UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

Marco Santos Souza

SIMULAÇÃO INTERATIVA DE TECIDOS E ROUPAS: TÉCNICAS PARA O DESENVOLVIMENTO DE UM SIMULADOR

Florianópolis (SC)

2014

Marco Santos Souza

SIMULAÇÃO INTERATIVA DE TECIDOS E ROUPAS: TÉCNICAS PARA O DESENVOLVIMENTO DE UM SIMULADOR

Dissertação submetida ao Programa de Pós-graduação em Ciência da Computação para a obtenção do Grau de Mestre em Ciências da Computação.
Orientador: Aldo von Wangenheim, Prof. Dr. rer. nat.

Florianópolis (SC)

2014

Catalogação na fonte elaborada pela biblioteca da
Universidade Federal de Santa Catarina

A ficha catalográfica é confeccionada pela Biblioteca Central.

Tamanho: 7cm x 12 cm

Fonte: Times New Roman 9,5

Maiores informações em:
http://www.bu.ufsc.br/design/Catalogacao.html

RESUMO

Há o interesse na simulação de roupas em qualquer aplicação onde há a presença de humanos virtuais. Contudo, simular o comportamento de tecidos no corpo de uma pessoa é uma tarefa complexa. Além disso, existe a constante necessidade de se obter resultados mais rápidos que viabilizem a simulação de indumentárias cada vez mais detalhadas. Este trabalho aborda técnicas relativas a diferentes etapas e aspectos do processo de simulação. Propõe-se uma otimização para um método para gerar efeitos de rasgo e corte. Ainda, introduz-se um algoritmo para a geração de linhas de costura em malhas de triângulos. As técnicas aqui descritas são utilizadas na implementação de um simulador cujo propósito é agilizar o desenvolvimento de sistemas para projeto e confecção de roupas.

Palavras-chave: Animação Baseada em Física, Simulação de Tecidos, Simulação de Rasgo, Detecção de Colisão

ABSTRACT

There is interest in the simulation of clothes in any application where there is the presence of virtual humans. However, simulate the behavior of fabric on the body of a person is a complex task. In addition, there is a constant need to get faster results that allow the simulation of increasingly detailed garment. This work discusses techniques for different stages and aspects of the simulation process. We propose an optimization for a method used to generate effects of tearing and cutting. Also, we introduce an algorithm to generate seam lines for triangle meshes. The techniques described herein are used in the implementation of a simulator whose purpose is to facilitate the development of systems for designing and making clothes.
Keywords: Physics-Based Animation, Cloth Simulation, Tearing Simulation, Collision Detection

SUMÁRIO

LISTA DE FIGURAS

1 INTRODUÇÃO

No contexto deste trabalho, o termo *simulação* diz respeito a utilização do computador para imitar o comportamento físico de algum fenômeno ou objeto. Nota-se que, com a análise do resultado de uma simulação, é possível fazer determinadas previsões a respeito do mundo real.

A necessidade de realizar tais previsões motivou, de certa forma, o desenvolvimento da computação. Segundo Cline (2002), já em meados do século XX, alguns dos primeiros computadores eletrônicos da história foram utilizados pelas forças armadas norte-americanas para simular a trajetória de projéteis. Desde então, conforme constata Mirtich (1996), a capacidade de prever o comportamento do universo (ou de parte dele) se mostrou uma das mais poderosas habilidades que a humanidade desenvolveu.

Com a evolução da tecnologia, nosso poder de previsão aumentou consideravelmente, viabilizando uma grande variedade de aplicações. Exemplos vão desde simulações do fluxo do ar atmosférico para prever condições meteorológicas até a simulação do movimento e formação de galáxias, estrelas e planetas. Em áreas como a engenharia, muitos sistemas têm sua viabilidade e corretude previamente testadas via simulação antes de serem construídos. Nesses casos, são desenvolvidos detalhados modelos matemáticos que visam convergência e grande precisão dos resultados. Os cálculos são geralmente realizados por supercomputadores, podendo levar de minutos a dias para terminarem (ERLEBEN, 2004). Aplicações desse tipo são tratadas pelo campo de estudo conhecido como *computação científica*.

Há uma variedade de casos, no entanto, onde a obtenção mais rápida dos resultados de uma simulação se faz necessária: jogos eletrônicos, projetos em robótica, sistemas de realidade virtual e softwares de animação, por exemplo. A fim de atender a essa demanda, são empregados algoritmos e técnicas que adotam simplificações dos modelos matemáticos e físicos, de forma a abrir mão de certa precisão. Contudo, visa-se ainda estabilidade e alguma precisão dos resultados; isto é, espera-se que a simulação ao menos pareça correta e seja agradável aos olhos humanos.

Segundo Erleben (2004), essa linha de pesquisa – subárea de *computação gráfica* – é hoje conhecida como modelagem ou *animação baseada em física* (*physics-based animation*), termo primeiramente utilizado em um curso de uma conferência de 1987 da ACM SIGGRAPH [1], "*Topics in Physically-Based Modeling*", organizado por Alan H. Barr.

Inserido no contexto da animação baseada em física, este trabalho aborda a simulação de um tipo específico de material aplicada a um propósito bem definido: simulação de tecidos para prever ou simular o comportamento de roupas no corpo de uma pessoa.

1.1 PROBLEMA E MOTIVAÇÃO

Profissionais da indústria têxtil e da moda sabem que o aspecto e o caimento final de uma peça de roupa dependem de inúmeros fatores, que vão do processo de fiação à confecção: tipo e quantidade de fibras, torção do fio, número de fios, corte, costura, etc. Contudo, a simulação de tecidos atualmente possível de ser realizada de forma interativa ainda está longe de poder contemplar todos esses detalhes. Por tal motivo, certas aplicações (como o sistema da Figura 5) podem se tornar frustrantes para determinados tipos de usuários.

Se a simulação pudesse ser melhorada ao ponto de gerar animações realistas em *tempo-real* para qualquer tipo de tecido e roupa, tal tecnologia encontraria aplicações que iriam impactar diretamente em vários aspectos da vida diária das pessoas (CHOI; KO, 2005a). Nota-se que, em qualquer aplicação onde há a presença de humanos virtuais, há também o interesse na simulação de roupas.

Outra questão que motiva essa linha de pesquisa é a necessidade de rodar sistemas que fazem uso de simulação física em *hardwares* modestos. Os computadores comuns têm capacidade de processamento muito aquém dos supercomputadores utilizados em áreas como a *computação científica*, e os consoles de *video game*, por exemplo, são geralmente ainda mais limitados. Por isso, conforme mencionado por Müller et al. (2007), a criação de novos métodos, adaptados para

[1] Association for Computing Machinery's Special Interest Group on Computer Graphics

necessidades específicas, é outra grande motivação para se realizar pesquisas na área de animação baseada em física. Segundo Cordier (2004), uma das maiores dificuldades na área de simulação interativa, ou de tempo-real, é achar um balanço adequado entre precisão e eficiência computacional.

1.2 OBJETIVOS

O objetivo geral deste trabalho é melhorar a simulação interativa de tecidos e roupas. Para tanto, objetiva-se também o desenvolvimento de um simulador onde poderão ser testados diferentes métodos e otimizações.

- Objetivos específicos:

1. Implementar e comparar alguns dos métodos presentes na literatura que permitam simular o movimento e as propriedades de diferentes tipos de tecidos.

2. Incorporar na simulação funcionalidades de detecção e tratamento de colisão entre corpos deformáveis e rígidos (e.g., tecido e manequim).

3. Viabilizar a simulação de rasgo de tecidos sem comprometer a interatividade da simulação.

4. Possibilitar a geração de costuras entre os diferentes pedaços de tecido, tornando possível a confecção de peças de roupa complexas.

5. Com o propósito de facilitar a experimentação e teste das diferentes técnicas, desenvolver um protótipo que:

   (a) Simule uma peça de roupa qualquer sobre o corpo de um manequim virtual;

   (b) Permita o usuário interagir com os tecidos simulados;

   (c) Viabilize a alteração de alguns dos parâmetros da simulação durante a sua execução.

1.3 ORGANIZAÇÃO

O levantamento bibliográfico inicial realizado nesta pesquisa seguiu as convenções de revisão sistemática definidas no relatório técnico de Kitchenham (2004). Com base nos resultados dessa revisão, elaborou-se o texto apresentado no capítulo 2.

O capítulo 3 discute dois métodos distintos para simulação de tecidos de propósito geral. As descrições dos sistemas de Baraff e Witkin (1998) e Müller et al. (2007), feitas nas seções 3.1 e 3.2, respectivamente, devem ser vistas como complementos das descrições originais. A ideia é auxiliar no entendimento das equações e facilitar a implementação desses sistemas.

O capítulo 4 trata de métodos de detecção de colisão. Na simulação de corpos deformáveis em geral, tal tarefa consome um tempo considerável de processamento. Assim, esse capítulo apresenta formas de tornar esse tempo não proibitivo aos propósitos deste trabalho.

O capítulo 5 aborda técnicas para a simulação de rasgo, corte e costura de tecidos. Um simulador com tais funcionalidades é especialmente interessante para aplicações CAD de projeto e confecção de roupas. Ainda, o capítulo comenta como foi implementada a geração eficiente de uma malha de triângulos com base em um polígono qualquer.

O capítulo 6 faz breves comentários a respeito das técnicas e ferramentas usadas na implementação do simulador e apresenta alguns dos protótipos que foram criados ao longo do processo de desenvolvimento.

Conclusões, contribuições e limitações deste trabalho são apresentadas no capítulo 7, onde, por fim, discorre-se também sobre possíveis trabalhos futuros.

2 REVISÃO BIBLIOGRÁFICA

Simulação de tecidos é um tópico de estudo interdisciplinar, envolvendo áreas como engenharia, física, matemática e computação. Não qualquer pedaço de pano, este trabalho visa a simulação de vestimentas – as quais podem ser compostas por uma ou mais peças de roupa – simuladas no corpo de um manequim virtual. Para tanto, precisa-se considerar dois aspectos distintos da simulação: dinâmica de tecidos (capítulo 3) e detecção de colisão (capítulo 4). Há diversos trabalhos que abordam tais assuntos separadamente. Detecção de colisão é um amplo e ativo campo de estudos por si só, bastante considerado na área de *geometria computacional*.

Este capítulo apresenta um breve histórico dos estudos realizados sobre simulação de tecidos e roupas na área de computação gráfica. Caso o leitor requeira uma visão mais aprofundada, o artigo de Choi e Ko (2005a) traz um levantamento dos principais trabalhos que contribuíram especificamente para o avanço na simulação de roupas e comenta sobre os principais problemas de pesquisa relacionados ao tema. Já Nealen et al. (2006) apresenta um *survey* sobre simulação de corpos deformáveis em geral, incluindo tecidos e roupas. Ainda, Müller et al. (2008) apresenta uma revisão didática de vários conceitos e trabalhos relevantes na área de animação baseada em física.

2.1 HISTÓRICO

Em meados dos anos 1980, simulação de tecidos passou a ser de interesse da comunidade de computação gráfica, dado que os avanços nas tecnologias de hardware começaram a viabilizar simulações com alguma interatividade. Deste então, foram realizados diversos trabalhos abordando os diferentes aspectos da simulação.

Terzopoulos et al. (1987) foi o primeiro a desenvolver um modelo físico para uso na simulação de tecidos. Até então, os modelos usados para simulação interativa de corpos deformáveis em geral eram passivos, i.e., os objetos simulados não interagiam entre si nem com forças externas.

Figura 1: Animações geradas por Carignan et al. (1992).

Carignan et al. (1992) aprimora o trabalho de Terzopoulos et al. (1987) com a adição de tratamento de colisão e o uso de uma componente de força de *damping* mais precisa, proposta por Platt e Barr (1988). Ainda, Carignan et al. (1992) são os primeiros a esboçar um método para simulação de roupas no corpo de um manequim virtual. Eles apresentam um sistema onde as peças de roupa são modeladas como polígonos em duas dimensões para depois serem costuradas e postas sobre o corpo de um manequim em três dimensões. O modelo do manequim realiza algumas sequências de animação predefinidas (e.g., caminhada, corrida) e a simulação do tecido é feita sobre o seu corpo em movimento. O resultado da simulação é então usado para a geração de uma sequência de quadros que compõem pequenos filmes de animação (Figura 1).

Volino e Thalmann (1995) apresentam uma abordagem completa para simulação de propósito geral de tecidos e objetos deformaveis com base em um sistema massa-mola. Eles tratam a detecção de colisão do tecido com outros objetos e também a *auto colisão*, i.e., a colisão do tecido com ele próprio.

Provot (1995) observa que os modelos físicos propostos até a ocasião são eficientes e realistas; contudo, quando tensionado, o tecido simulado geralmente se comporta mais como borracha do que como tecido de fato. Provot utiliza então um modelo massa-mola com um pós-processamento para corrigir a posição de partículas conectadas por molas que excedem em demasia seu comprimento original (Figura 2). Posteriormente, o autor complementa o seu trabalho em um artigo

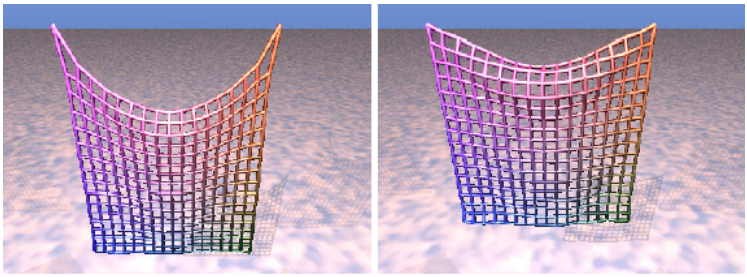

Figura 2: Simulação de tecido sem (a) e com a técnica de correção (b) sugerida por Provot (1995).

(PROVOT, 1997) que foca especificamente nos problemas de detecção e tratamento de colisão para o sistema proposto.

Baraff e Witkin (1998) afirmam que o gargalo na simulação de tecidos é oriundo do fato de que, para evitar instabilidades, o passo de tempo usado na simulação deve ser pequeno. Eles descrevem um método de integração implícita capaz de utilizar passos de tempo maiores sem grande perda de estabilidade, melhorando assim o desempenho da simulação (Figura 3). O método requer a resolução de um sistema linear, o que os autores sugerem que seja feito montando uma grande matriz esparsa e aplicando uma versão modificada do *método do gradiente conjugado* (SHEWCHUK, 1994), chamado de *modified preconditioned conjugate gradient*, ou *MPCG*, na literatura em inglês. Infelizmente, isso gera limitações para o uso em aplicações interativas. Outro problema é que o método do gradiente conjugado requer que a matriz de entrada seja simétrica e *positiva definida*, caso contrário o método pode não convergir, gerando problemas de instabilidades na simulação. Para algumas configurações do tecido, instabilidades de fato ocorrem. Contudo, trabalhos posteriores resolvem esse problema, melhorando a eficiência e a robustez do *MPCG* original (ASCHER; BOXERMAN, 2003). Muitos sistemas utilizam ou se baseiam no método de Baraff e Witkin em virtude de sua acurácia. A seção 3.1 traz mais detalhes a respeito.

Os trabalhos de Cordier e Magnenat-Thalmann (2002), Cordier (2004) são os primeiros a focar na simulação *interativa* (ou *real-time*)

de manequins virtuais vestindo diferentes tipos de roupa. A hipótese dos autores é de que não há a necessidade de se simular a roupa inteira utilizando um método de simulação de proposito geral. Ao invés disso, muitas simplificações podem ser feitas com base no tipo de roupa e a maneira como cada uma de suas diferentes partes interagem com o corpo que as veste.

Figura 3: Simulação de tecidos com o método de integração implícita sugerido por Baraff e Witkin (1998).

Baraff, Witkin e Kass (2003) constantam que o tratamento de colisão de tecido com outro tecido (ou com ele mesmo) é uma das maiores deficiências na maioria dos simuladores. Outro problema é o tratamento de colisão feito quando o tecido é simulado no corpo de um manequim animado, e algumas partes do manequim podem se intersectar, colocando a roupa em um estado difícil de tratar. Os autores apresentam algoritmos para amenizar tais questões.

Choi e Ko (2005b) observam que a instabilidade encontrada em alguns métodos de simulação de tecidos é proveniente do fenômeno da flambagem, presente também, de forma diminuta, em tecidos, mas erroneamente não considerado na formulação dos modelos matemáticos usados até então. Eles apresentam uma técnica de integração semi-implícita que, levando em consideração a existência da flambagem, é capaz de produzir resultados mais estáveis dos que o de Baraff e Witkin (1998), por exemplo.

Com o foco em aplicações interativas e jogos eletrônicos, Müller et al. (2007) propõem uma técnica diferenciada para a simulação de

tecidos e corpos deformáveis em geral, chamada de *position based dynamics* (*PBD*), na literatura em inglês (Figura 4). A ideia é manipular diretamente a posição das partículas envolvidas em cada restrição de movimento estabelecida ao invés da aplicação de forças. Alterações na velocidade de cada partícula são feitas explicitamente com base no deslocamento aplicado. O método é atraente devido a sua eficiência, estabilidade, facilidade de implementação e uso. Expansões e otimizações são propostas em trabalhos posteriores (MüLLER, 2008; MüLLER; CHENTANEZ, 2010). A seção 3.2 discorre sobre o trabalho de Müller et al. (2007).

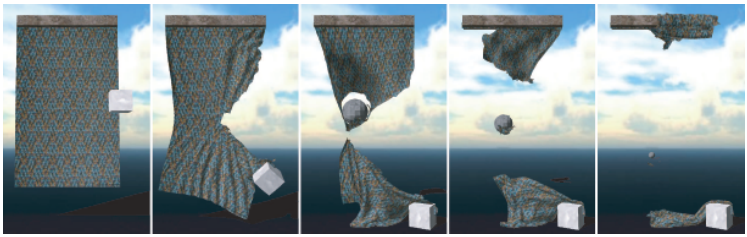

Figura 4: Simulação interativa de tecido sendo rasgado usando o método proposto por Müller et al. (2007).

Com relação ao rasgo de tecidos, Metaaphanon et al. (2009) propõem um modelo de dois níveis para simular rasgo ou corte realisticamente mostrando fios soltos na malha, que geralmente aparecem quando um tecido de algodão, por exemplo, é rasgado. A ideia é representar o tecido por fios e tramas individuais, como ocorre em um tecido real, e não como um material inteiro único, como é feito na maioria dos modelos usados em computação gráfica. Essa abordagem, porém, está longe de atingir resultados em tempo real. Nos testes realizados pelos autores, são necessários vários segundos para simular um único quadro.

Bender, Weber e Diziol (2013) apresentam um método para simulação de tecido baseado na técnica conhecida como *shape matching*. O resultado é uma simulação eficiente e estável; contudo, o método é capaz de representar apenas uma variedade limitada de tecidos. Os autores mostram também como a técnica pode ser otimizada com o uso da *GPU* (*graphics process unit*).

2.2 ESTADO DA ARTE

Visando simular vestimentas mais complexas em tempo real, trabalhos recentes buscam explorar pontos passiveis de paralelização nos métodos de simulação e detecção de colisão já existentes (TANG et al., 2013; SCHMITT et al., 2013). Isso ocorre devido à popularização dos processadores *multicore* – presentes em diferentes dispositivos – e, também, pela disseminação das placas gráficas e a evolução das ferramentas para as linguagens de programação de propósito geral para *GPU* (e.g., *CUDA* e *OpenCL*).

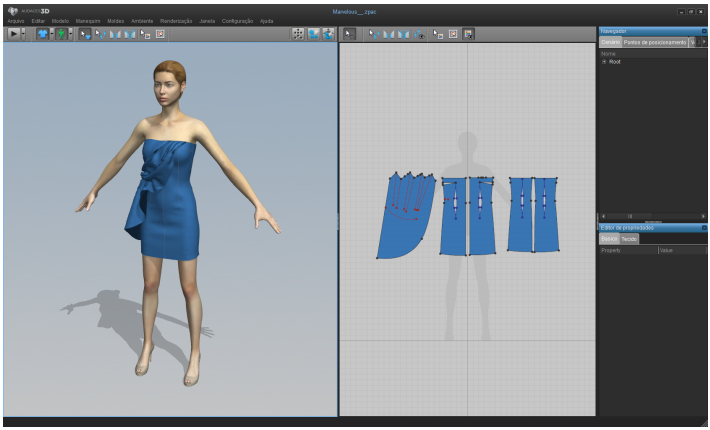

Figura 5: Exemplo de software CAD usado no projeto de roupas.

De maneira geral, as pesquisas em simulação de tecidos na área de computação gráfica – combinadas com os avanços nas capacidades de hardware – tornaram viável o uso dessa tecnologia para diversos propósitos. Contudo, não existe uma melhor abordagem para todos os casos. Cada aplicação precisa escolher e adaptar as técnicas existentes de acordo com suas necessidades. Nota-se que, para certos fins, é possível obter resultados satisfatórios (e.g., jogos); para outros, porém, a simulação pode gerar resultados frustrantes (e.g., sistemas para projeto de roupas, Figura 5), conforme comentado na seção 1.1.

3 DINÂMICA DE TECIDOS

A maneira utilizada para representar computacionalmente a superfície de um tecido é um ponto inicial a se considerar. Apesar de haver abordagens que adotam o *Método dos Elementos Finitos* (COLLIER et al., 1991; EISCHEN; DENG; CLAPP, 1996), uma representação com sistema de partículas é, aparentemente, a única opção viável para aplicações interativas.

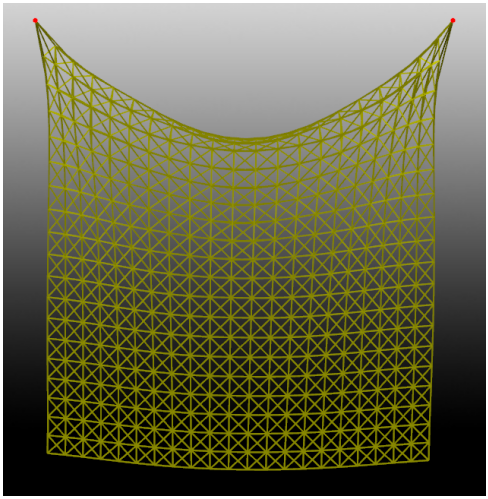

Figura 6: Exemplo de tecido simulado com um sistema de partículas do tipo massa-mola. Cada aresta da malha representa restrições de movimento entre as partículas do sistema.

Em um sistemas de partículas, um tecido é representado por um conjunto de partículas cujas posições iniciais são provenientes dos vértices de uma malha poligonal (e.g., uma malha de triângulos) e, também, por um conjunto de restrições (*constraints*, na literatura em inglês) sobre o movimento dessas partículas. Tais restrições são calculadas possivelmente com base nas relações geométricas que existem entre os vértices na malha poligonal (Figura 6). Contudo, o modo como tais restrições são de fato criadas e aplicadas, a forma como as

partículas são deslocadas em um intervalo de tempo (i.e., o método de integração) e a maneira como colisões são detectadas e tratadas são alguns dos pontos que variam entre os sistemas existentes na literatura. Este capítulo discorre sobre dois deles.

Independente do sistema usado, cada partícula tem seu movimento suficientemente descrito pela mecânica newtoniana. Assim, a variação de posição (e velocidade) de cada partícula pode ser calculada partindo-se de

$$\ddot{\mathbf{x}} = \mathbf{M}^{-1}\left(-\frac{\partial E}{\partial \mathbf{x}} + \mathbf{f}_{ext}\right), \tag{3.1}$$

que é, basicamente, $\sum \mathbf{F} = m\ddot{\mathbf{x}}$ (da segunda lei de Newton) reescrita. $\mathbf{M}$ é a matriz diagonal de ordem três, $\mathbf{M} = diag(m,m,m)$, onde m é a massa da partícula e $\mathbf{x}$ sua posição. O termo $\left(-\frac{\partial E}{\partial \mathbf{x}} + \mathbf{f}_{ext}\right)$ representa o vetor de forças resultantes atuando na partícula, sendo que $\mathbf{f}_{ext}$ contém as forças externas (vento, colisão, atrito, etc) e E é uma função escalar de $\mathbf{x}$ que descreve a energia interna do tecido – ou as forças internas, que são resultantes das restrições de movimento estabelecidas entre as partículas do sistema. Segundo Baraff e Witkin (1998), o comportamento interno de tecidos é costumeiramente descrito em termos desse tipo de função. A força $\mathbf{f}_{int}$ gerada por essa energia é dada pela negação da derivada parcial de $E(\mathbf{x})$ em relação a $\mathbf{x}$, isto é,

$$\mathbf{f}_{int} = -\frac{\partial E}{\partial \mathbf{x}}. \tag{3.2}$$

Para ser usada computacionalmente, a equação 3.1 é reescrita como uma equação diferencial ordinária (EDO) e discretizada. Dessa maneira, é possível aplicar um método de integração numérica para calcular novos valores de velocidade e posição.

Um dos métodos mais populares é o método de Euler, que é classificado como um método *explícito* de primeira ordem. O método pode ser descrito pelas seguintes equações:

$$\mathbf{v}^{t+h} = \mathbf{v} + h(\mathbf{M}^{-1}\mathbf{f}), \tag{3.3}$$

$$\mathbf{x}^{t+h} = \mathbf{x} + \mathbf{v}^{t+h}h,$$

onde h é um passo de tempo escolhido, $\mathbf{v}$ a velocidade e $\mathbf{x}$ a posição da partícula no tempo t. $\mathbf{f}$ é um vetor que representa todas as forças (internas e externas) atuando na partícula no tempo t. O método é chamado de "explícito" porque $\mathbf{x}^{t+h}$ (a posição no tempo seguinte, $t+h$) pode ser escrito explicitamente em função de $\mathbf{x}$ e $\mathbf{v}$ (posição e velocidade no tempo atual, t).

A fim de simplificar a notação usada a seguir, define-se:

$$\mathbf{x} \in \mathbb{R}^{3n} = [\mathbf{x}_1, \mathbf{x}_2, ...\mathbf{x}_n]^T ,$$

onde $\mathbf{x}_i \in \mathbb{R}^3$ é a posição da i-ésima partícula. Ou seja, $\mathbf{x}$ foi redefinido como sendo a matriz coluna que contém as posições de todas as n partículas do sistema. Analogamente, $\mathbf{v} \in \mathbb{R}^{3n}$ agora é a matriz de velocidades e $\mathbf{f} \in \mathbb{R}^{3n}$ a de forças. A matriz $\mathbf{M}$, chamada de *matriz de inércia*, passa a conter em sua diagonal as massas de todo o sistema, isto é, $\mathbf{M} \in \mathbb{R}^{3n \times 3n} = diag(m_1, m_1, m_1, m_2, m_2, m_2, ..., m_n, m_n, m_n)$.

Por último, define-se $\Delta\mathbf{x}$ e $\Delta\mathbf{v}$ como a diferença entre as posições e velocidades, respectivamente, após um intervalo de tempo h:

$$\Delta\mathbf{x} = \mathbf{x}^{t+h} - \mathbf{x},$$

$$\Delta\mathbf{v} = \mathbf{v}^{t+h} - \mathbf{v}.$$

3.1 SISTEMA DE BARAFF E WITKIN (1998)

Métodos de integração *explícita* são comumente usados na simulação física de corpos rígidos (SOUZA, 2011), por exemplo. Contudo, Baraff e Witkin (1998) argumentam que esses métodos não são adequados para a simulação de tecidos porque, para esse tipo de material, requerem um passo de tempo muito pequeno para avançar a simulação sem perda de estabilidade. Isso, na prática, limita o uso dos métodos explícitos em aplicações interativas. Métodos de integração *implícita*, entretanto, são capazes de usar passos de tempo maiores.

O método de Euler *implícito* pode ser descrito pelas seguintes equações:

$$\mathbf{v}^{t+h} = \mathbf{v} + h(\mathbf{M}^{-1}\mathbf{f}^{t+h}), \tag{3.4}$$

$$\mathbf{x}^{t+h} = \mathbf{x} + \mathbf{v}^{t+h}h. \tag{3.5}$$

Comparando as equações das duas versões do método de Euler (explícita e implícita), nota-se que a única diferença está no uso de $\mathbf{f}^{t+h}$ na equação 3.4 ao invés de $\mathbf{f}$ (equação 3.3). $\mathbf{f}^{t+h}$, o vetor de forças no próximo intervalo de tempo, tem valor desconhecido, contudo, pode ser definido em função de $\mathbf{f}$ com

$$\mathbf{f}^{t+h} = \mathbf{f} + \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \Delta \mathbf{x}.$$

Com isso, realizando algumas substituições e derivações, Baraff e Witkin (1998) mostram como a equação 3.4 dá origem a um sistema linear onde $\Delta\mathbf{v}$ é o valor a ser calculado. Ou seja, enquanto que na versão explícita do método de Euler o valor de $\Delta\mathbf{v}$ é trivialmente obtido, na versão implícita é preciso resolver um sistema linear. O sistema é

$$\left(\mathbf{I} - h^2\mathbf{M}^{-1}\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right)\Delta\mathbf{v} = h\mathbf{M}^{-1}\left(\mathbf{f} + h\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\mathbf{v}\right), \tag{3.6}$$

onde $\mathbf{I}$ é a matriz identidade de tamanho $3n \times 3n$, sendo n o número de partículas do sistema.

A equação 3.6 está no formato

$$\mathbf{Az} = \mathbf{b}, \tag{3.7}$$

notação matricial usada em sistemas lineares onde, genericamente, $\mathbf{A}$ é uma matriz de tamanho $j \times k$, $\mathbf{z}$ é uma matriz coluna com k elementos e $\mathbf{b}$ é uma matriz coluna com j elementos. $\mathbf{A}$ e $\mathbf{b}$ têm valores conhecidos e $\mathbf{z}$ deve ser calculado de forma a satisfazer a equação 3.7. No caso da equação 3.6, tem-se que $\mathbf{A} = \left(\mathbf{I} - h^2\mathbf{M}^{-1}\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right)$ é uma matriz de tamanho $3n \times 3n$, $\mathbf{b} = h\mathbf{M}^{-1}\left(\mathbf{f} + h\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\mathbf{v}\right)$ e $\mathbf{z} = \Delta\mathbf{v}$ são ambas matrizes coluna com $3n$ elementos.

Baraff e Witkin (1998) utilizam o *método do gradiente conjugado* para achar os valores de $\Delta\mathbf{v}$ que satisfazem o sistema da equação 3.6. Assim, $\mathbf{v}^{t+h}$ é obtido por $\mathbf{v}^{t+h} = \mathbf{v} + \Delta\mathbf{v}$ e, com isso, $\mathbf{x}^{t+h}$ é trivialmente calculado pela equação 3.5.

O método do gradiente conjugado (ou *conjugate gradient*, *CG*, na literatura em inglês) é um método iterativo bastante usado na re-

solução de grandes sistemas de equações lineares. Uma detalhada descrição é fornecida por Shewchuk (1994). O método é aplicado a sistemas no formato da equação 3.7, sendo que a matriz **A** precisa ser simétrica e definida positiva. Na formulação proposta por Baraff e Witkin (1998), a matriz **A** não tem garantia de ser definida positiva. Contudo, os autores adicionam uma força de *damping* projetada para aumentar a probabilidade de **A** ser definida positiva. Infelizmente, não há garantia de que isso sempre ocorra. Na prática, o método diverge para certas configurações do tecido, o que gera instabilidades na simulação. Felizmente, esses problemas são contornados por trabalhos posteriores (ASCHER; BOXERMAN, 2003; CHOI; KO, 2005b).

A subseção seguinte apresenta uma forma de calcular as forças, $\mathbf{f}$, e as derivadas parciais, $\frac{\partial \mathbf{f}}{\partial \mathbf{x}}$, presentes na equação 3.6. Baraff e Witkin (1998) apresentam uma maneira diferente, onde as forças são calculadas baseando-se na deformação de cada triângulo da malha. Macri (2000) e Pritchard (2003) apresentam trabalhos que podem auxiliar na implementação das equações originais fornecidas por Baraff e Witkin.

3.1.1 Cálculo de Forças Internas

As forças internas atuantes em cada partícula podem ser calculadas usando um simples modelo massa-mola. Nesse modelo, a função de energia $E(x)$ que descreve a deformação entre duas partículas i e j é dada por

$$E = \begin{cases} \frac{1}{2}k_s(\left|\mathbf{x}_{ij}\right| - L)^2 & : \quad \left|\mathbf{x}_{ij}\right| \geq L \\ 0 & : \quad \left|\mathbf{x}_{ij}\right| < L \end{cases}$$

onde $\mathbf{x}_{ij} = \mathbf{x}_j - \mathbf{x}_i$ (vetor entre as duas partículas), L é a distância original entre as partículas e k_s é a constante elástica da mola. Segundo a equação 3.2, a força atuante na partícula i gerada por essa energia pode ser calculada com

$$\mathbf{f}_i = -\frac{\partial E}{\partial \mathbf{x}} = \begin{cases} k_s(\left|\mathbf{x}_{ij}\right| - L)\frac{\mathbf{x}_{ij}}{\left|\mathbf{x}_{ij}\right|} & : \quad \left|\mathbf{x}_{ij}\right| \geq L \\ 0 & : \quad \left|\mathbf{x}_{ij}\right| < L \end{cases}$$

A equação 3.6 requer ainda a derivada parcial de f em relação a x, que pode ser calcula por

$$\frac{\partial \mathbf{f}_i}{\partial \mathbf{x}_j} = \begin{cases} k_s \frac{\mathbf{x}_{ij}\mathbf{x}_{ij}^T}{\mathbf{x}_{ij}^T\mathbf{x}_{ij}} + k_s \left(1 - \frac{L}{|\mathbf{x}_{ij}|}\right)\left(\mathbf{I} - \frac{\mathbf{x}_{ij}\mathbf{x}_{ij}^T}{\mathbf{x}_{ij}^T\mathbf{x}_{ij}}\right) & : \quad |\mathbf{x}_{ij}| \geq L \\ 0 & : \quad |\mathbf{x}_{ij}| < L \end{cases}$$

3.2 SISTEMA DE MÜLLER ET AL. (2007)

Müller et al. (2007) propõem uma abordagem de simulação ligeiramente diferente das existentes até então. Na maioria dos sistemas de partículas, forças externas e internas são calculadas e acumuladas para obter-se a aceleração de cada partícula com base na segunda lei de Newton (equação 3.1). Depois, um método de integração é usado para computar novos valores de velocidade e posição (e.g., método de Euler implícito, equações 3.4 e 3.5).

A abordagem é chamada pelos autores de "dinâmica baseada em posição" – *position based dynamics*, PBD. Ao invés de calcular acelerações, velocidades e, por fim, posições, a idéia é manipular diretamente as posições das partículas simuladas. Os autores observam que alguns métodos de simulação, classificados como *métodos baseados em impulso*, não calculam a aceleração usando a equação 3.1, mas sim alteram diretamente a velocidade[1]. No método PBD, a "camada de velocidade" também é omitida.

A técnica é aplicável na simulação de objetos deformáveis em geral, não somente tecidos. Um objeto é representado por um conjunto de n vértices (partículas) e N restrições sobre o movimento dos mesmos. O algoritmo 3.1, fornecido por Müller et al. (2007), ilustra a abordagem. A notação utilizada é semelhante a apresentada até aqui neste capítulo.

As linhas de 1 a 3 apenas definem o estado inicial do objeto simulado. Considerando apenas as forças externas, as linhas de 5 a 7 aplicam o método de integração de Euler explícito para obter uma

[1] "Impulso" é definido como uma *alteração instantânea na velocidade*. Sistemas baseados em impulso são geralmente usados na simulação de corpos rígidos (SOUZA, 2011), onde as forças geradas por colisões, por exemplo, são aplicadas via alterações nas velocidades dos objetos em contato.

estimativa **p** da posição de cada vértice i. A linha 6 chama uma rotina que aplica *damping* nas velocidades antes de elas serem usadas para calcular as estimativas de posição na linha 7.

Algoritmo 3.1 Simulação com *PBD* (MüLLER et al., 2007).

```
1: forall vertices i
2: .     initialize x_i = x_i^0, v_i = v_i^0, w_i = 1/m_i
3: endfor
4: loop
5: .     forall vertices i do v_i ← v_i + h w_i f_ext(x_i)
6: .     dampVelocities (v_i, ..., v_n)
7: .     forall vertices i do p_i ← x_i + h v_i
8: .     forall vertices i do generateCollisionConstraints(x_i → p_i)
9: .     loop solverIterations times
10: .    .     projectConstraints(C_1, ..., C_{N+N_col}, p_1, ..., p_n)
11: .    endloop
12: .    forall vertices i do
13: .    .     v_i ← (p_i − x_i)/h
14: .    .     x_i ← p_i
15: .    endfor
16: .    velocityUpdate(v_1, ..., v_n)
17: endloop
```

Na linha 8, a diferença entre a posição atual **x** e a estimativa **p** é usada para detectar possíveis colisões, gerando N_{col} novas restrições de movimento. Observa-se que existem N restrições inicias fixas (restrições internas) mais N_{col} restrições que são geradas dinamicamente a cada iteração.

Entre as linhas 9 e 11, as $N + N_{col}$ restrições são usadas para corrigir as n estimativas de posição **p**. Ou seja, para cada restrição, os vértices envolvidos são deslocados a fim de satisfazê-la. Os autores chamam o processo de aplicar uma restrição de "projeção de restrição". Cada restrição é projetada independentemente das demais. Contudo, ao projetar uma restrição, pode-se estar alterando o resultado das demais. A fim de minimizar tal efeito, o processo é repetido por um número arbitrário de vezes (*solverIterations*, na linha 9). Ressalta-se que a projeção de restrições trabalha apenas sobre as estimativas

de posição $\mathbf{p}_i$. As posições originais, $\mathbf{x}_i$, são mantidas intactas neste processo.

Entre as linhas 12 e 15, as velocidades são ajustadas com base no deslocamento real, $\mathbf{x}_i \rightarrow \mathbf{p}_i$, realizado por cada vértice i no intervalo de tempo h. Em seguida, as posições $\mathbf{x}$ são setadas com o valor das estimativas $\mathbf{p}$. Por fim, na linha 16, apenas as velocidades dos vértices que tiveram restrições de colisão geradas são ajustadas de acordo com os coeficientes de atrito e restituição, que são propriedades pré-definidas do material simulado.

3.2.1 Projetando Restrições

Resta definir como implementar o passo de "projetar restrições", executado na linha 10 do algoritmo 3.1.

Basicamente, uma restrição j, $j \in [1,...,N]$, possui uma cardinalidade n_j, que indica quantas partículas ela afeta, e uma função $C_j : \mathbb{R}^{3n_j} \rightarrow \mathbb{R}$. Essa função é usada para calcular o valor de deslocamento, $\Delta\mathbf{p}_i$, que deve ser aplicado a cada partícula envolvida:

$$\Delta\mathbf{p}_i = -sw_i\nabla_{\mathbf{p}_i}C(\mathbf{p}_1,...,\mathbf{p}_n),$$

onde w_i é o inveso da massa da partícula i, $w_i = 1/m_i$, e s é um valor escalar, igual para todas as partículas da restrição, definido por:

$$s = \frac{C(\mathbf{p}_1,...,\mathbf{p}_n)}{\sum_j w_j \left|\nabla_{\mathbf{p}_j}C(\mathbf{p}_1,...,\mathbf{p}_n)\right|^2}$$

Por exemplo, a função C da restrição de distância entre as posições $\mathbf{p}$ (estimativas) de duas partículas quaisquer pode ser definida por $C(\mathbf{p}_1,\mathbf{p}_2) = |\mathbf{p}_1 - \mathbf{p}_2| - d$, onde d é a distância original entre as partículas. Os gradientes para cada posição são dados por $\nabla_{\mathbf{p}_1}C(\mathbf{p}_1,\mathbf{p}_2) = \mathbf{n}$ e $\nabla_{\mathbf{p}_2}C(\mathbf{p}_1,\mathbf{p}_2) = -\mathbf{n}$, onde $\mathbf{n} = \frac{\mathbf{p}_1-\mathbf{p}_2}{|\mathbf{p}_1-\mathbf{p}_2|}$. Assim, $s = \frac{|\mathbf{p}_1-\mathbf{p}_2|-d}{w_1+w_2}$ e as correções de posição são

$$\Delta\mathbf{p}_1 = -\frac{w_1}{w_1+w_2}(|\mathbf{p}_1-\mathbf{p}_2|-d)\frac{\mathbf{p}_1-\mathbf{p}_2}{|\mathbf{p}_1-\mathbf{p}_2|},$$

$$\Delta \mathbf{p}_2 = +\frac{w_2}{w_1 + w_2}(|\mathbf{p}_1 - \mathbf{p}_2| - d)\frac{\mathbf{p}_1 - \mathbf{p}_2}{|\mathbf{p}_1 - \mathbf{p}_2|}.$$

Müller et al. (2007) mostram como calcular os valores de $\Delta \mathbf{p}$ para outros tipos de restrições comumente usadas na simulação de tecidos, como restrições de dobra (*bending*), que envolve quatro partículas.

4 DETECÇÃO DE COLISÃO

Segundo Ericson (2005), *detecção de colisão* é um tópico amplo que trata de um problema aparentemente simples: dizer *se*, *quando* e *como* dois (ou mais) objetos estão em contato. Geralmente, tal tarefa consome um tempo considerável de processamento. As técnicas mostradas neste capítulo visam tornar esse tempo não proibitivo aos propósitos deste trabalho.

A abordagem comumente utilizada é dividir o processo de detecção de colisão em duas fases distintas, que são executadas em sequência. A primeira é chamada de "fase ampla" (*broad phase*, seção 4.1); a segunda, de "fase restrita" (*narrow phase*, seção 4.2).

4.1 FASE AMPLA

O objetivo da fase ampla é diminuir o número de pares de objetos que precisarão ser testados na fase restrita. No simulador desenvolvido neste trabalho, a fase ampla é utilizada genericamente para limitar o número de testes realizados entre diferentes peças de tecido e também entre corpos rígidos. O processo realizado na fase ampla é também conhecido na literatura por *collision culling*.

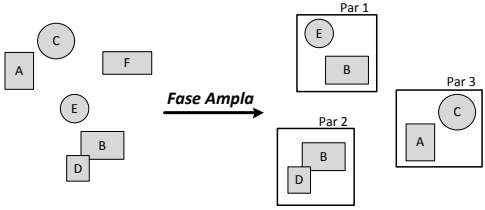

Figura 7: Exemplo onde a fase ampla do processo de detecção de colisão encontrou três pares de objetos que estão possivelmente colidindo.

A Figura 7 exemplifica o papel desta fase. De alguma maneira, detectou-se três pares de objetos em possível colisão, apesar de que, nesse caso, apenas o par {B, D} representa realmente uma colisão.

Isso só será descoberto, entretanto, na fase restrita.

Idealmente, a fase ampla não seria necessária, pois a detecção de colisão é de fato realizada na fase restrita. Entretanto, as técnicas empregadas na fase restrita podem ser computacionalmente custosas, de forma que realizá-las sobre cada par de objetos presente na simulação se torna inviável a medida que a quantidade de objetos aumenta. Em uma simulação com n objetos, por exemplo, testar todos os pares – ou seja, uma abordagem de *força bruta* – resulta em $n(n-1)/2$ testes. Isso representa uma complexidade temporal de $O(n^2)$. Portanto, reduzir o custo associado a cada teste é apenas uma solução paliativa para valores de n relativamente pequenos.

A subseção 4.1.1 mostra como reduzir o custo associado a cada teste. As subseções 4.1.2 e 4.1.3 mostram algoritmos com uma melhor complexidade temporal, i.e., que visam reduzir o número de testes.

4.1.1 *Bounding Volumes*

A fim de agilizar os testes realizados na fase ampla, cada objeto é envolto por uma forma geométrica suficientemente simples, que tenha o menor volume possível mas ainda sim contenha *toda* sua geometria. Essa forma simples é chamada de *bounding volume* e geralmente se trata de uma esfera, um paralelepípedo reto sem orientação espacial – conhecido como *axis-aligned bounding box* ou somente AABB – ou então com orientação, chamado de *oriented bounding box*, ou OBB.

Verificar se os *bounding volumes* de dois objetos se intersectam é uma tarefa computacionalmente eficiente, e deve preceder um teste de intersecção mais complexo.

Há um *trade-off* entre ajuste à geometria e velocidade de teste de intersecção a ser considerado para essas três formas, conforme é ilustrado pela Figura 8. Uma OBB geralmente se encaixa mais facilmente na geometria do objeto. Porém, realizar testes de intersecção com OBBs não é tão simples como com esferas que, por sua vez, fornecem uma aproximação demasiadamente grosseira da geometria na maioria dos casos.

No simulador desenvolvido neste trabalho, qualquer teste na fase ampla é feito utilizando apenas os *bounding volumes*. Isso vale,

inclusive, para os dois algoritmos apresentados nas subseções as seguir (4.1.2 e 4.1.3). Dessa maneira, a fase ampla não precisa conhecer a real geometria de cada objeto, apenas seus *bounding volumes*.

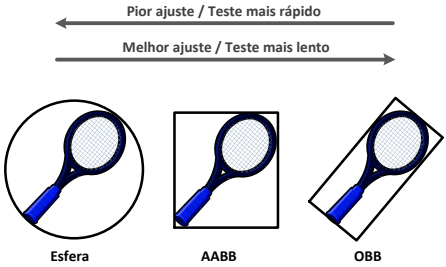

Figura 8: Três tipos de *bounding volume* e o *trade-off* entre ajuste à geometria e velocidade de teste de intersecção existente entre eles.

4.1.2 Grade Uniforme

A técnica de grade uniforme é simples e, talvez, uma das mais adotadas na fase ampla do processo de detecção de colisão. Há diversas variações e maneiras de tornar essa abordagem mais eficiente para casos específicos. Para um estudo mais aprofundado, indica-se os trabalhos de Bergen (BERGEN, 2003) ou Ericson (ERICSON, 2005). O artigo de Scott Le Grand, presente no livro editado por Hubert Nguyen (NGUYEN, 2007), apresenta uma forma de paralelizar essa técnica e implementa-lá em GPU.

Trata-se de um método de subdivisão espacial. A ideia consiste em se ter uma grade que sobrepõe o espaço no qual os objetos podem colidir. Essa grade é composta de várias células (ou regiões) de mesmo tamanho, que particionam o espaço de maneira uniforme.

A Figura 9 ilustra uma grade e os objetos em uma determinada distribuição. A ilustração é feita em um espaço bidimensional apenas para facilitar a visualização; o funcionamento é perfeitamente análogo em três dimensões. Nota-se que um objeto pode ocupar o espaço de mais de uma célula ao mesmo tempo (i.e., mais de uma célula pode conter o mesmo objeto). O objeto A, por exemplo, está em quatro

células, enquanto que o objeto F está em apenas uma.

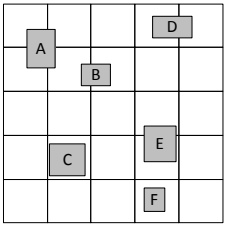

Figura 9: Grade uniforme bidimensional sobrepondo o espaço onde seis objetos (rotulados de A a F) estão distribuídos.

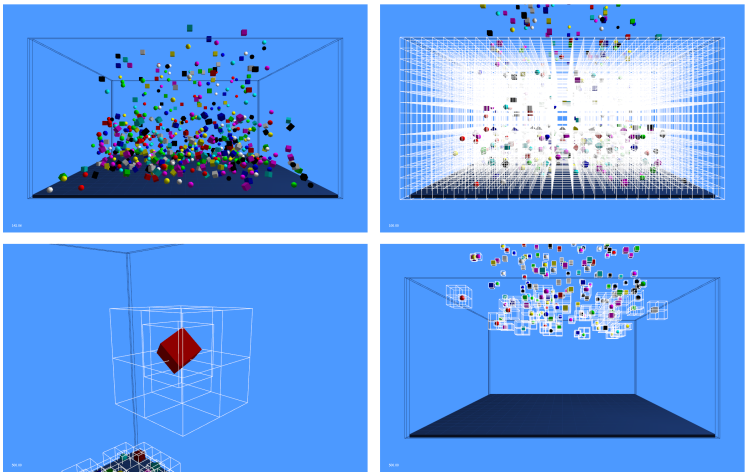

Figura 10: *Screenshots* da aplicação usada para testes do método da grade uniforme. Na imagem superior direita toda a grade é mostrada, enquanto que nas duas imagens inferiores são mostradas apenas as células que contêm algum objeto.

Dois objetos podem colidir se e somente se eles ocuparem as mesmas células. Assim, para descobrir com quais outros objetos um determinado objeto pode estar colidindo, basta testá-lo com os objetos

que estão nas mesmas células ocupadas por ele. Geralmente se estabelece que o tamanho de cada célula seja suficientemente grande para conter totalmente o maior dos volumes presentes na simulação. Com isso, um objeto poderá estar no máximo em oito células ao mesmo tempo (ou quatro, no caso bidimensional).

A Figura 10 mostra um protótipo que foi desenvolvido para testar o método da grade uniforme na simulação de corpos rígidos. No cenário simulado, vários objetos rígidos, no formato de cubos ou esferas, estão inicialmente suspensos em posições aleatórias e movem-se sob ação da gravidade, estando aptos a colidirem entre si e com as paredes do ambiente virtual.

4.1.3 *Sweep and Prune* (SAP)

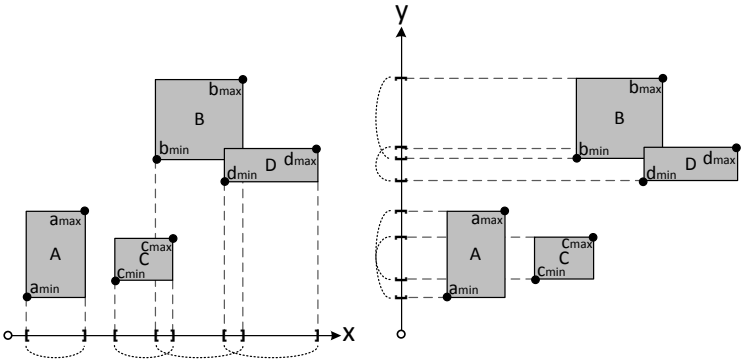

Figura 11: Exemplo em duas dimensões ilustrando as AABBs de quatro objetos (*A*, *B*, *C* e *D*). À esquerda está a projeção das AABBs no eixo *x* e, a direita, no eixo *y*.

Os métodos de subdivisão espacial (como o da grade uniforme, subseção 4.1.2) precisam lidar com algumas complicações, como as que são decorrentes do fato que objetos podem ocupar mais de uma região (ou célula) ao mesmo tempo (ERICSON, 2005). Ao invés de manter os objetos em uma estrutura fixa complexa como grades ou

octrees, o método conhecido por *sweep and prune* (SAP)[1] mantém os objetos em simples listas encadeadas que são ordenadas pela posição de cada objeto. A maior vantagem do SAP, porém, é que ele é capaz de explorar a coerência que existe com relação a posição dos objetos entre consecutivos quadros em uma simulação física. Ou seja, ele explora o fato de que o que era válido no quadro anterior muito provavelmente ainda é valido, ou está próximo de ser válido, no quadro atual.

Uma diferença importante com relação ao método da grade uniforme é que o SAP não encontra em apenas um único quadro da simulação todos os pares que podem representar uma colisão. O trabalho é feito de forma incremental, isto é, a cada novo quadro, novos pares são encontrados ou descartados. Por isso, é preciso manter um registro de todos os pares que podem representar uma colisão. A seguir, mostra-se como o SAP atualiza esse registro.

A Figura 11 ilustra um cenário bidimensional onde quatro objetos têm os extremos de seus *bounding volumes* projetados nos eixos x e y (o funcionamento do SAP em três dimensões é perfeitamente análogo). Assume-se aqui que todos os *bounding volumes* são AABBs (subseção 4.1.1), pois essa forma geométrica já guarda os valores extremos em cada eixo, evitando-se assim qualquer cálculo para obter as projeções.

Pelo teorema de separação nos eixos (*separating axis theorem*, SAT), sabe-se que, caso o intervalo das projeções de duas AABBs se intersectem em ambos os eixos x e y (como ocorre com os objetos B e D), as próprias AABBs se intersectam.

Cada eixo na Figura 11 está associado a uma lista. Em cada lista serão inseridos os valores extremos das projeções das AABBs no eixo associado. Um elemento dessas listas precisa conter, essencialmente, quatro informações. A primeira é uma referência para uma AABB. A segunda é uma *flag* (valor booleano) para indicar se o elemento representa o valor extremo máximo ou o mínimo da projeção da AABB no eixo associado à lista em questão. A terceira e quarta são duas referências, uma para o próximo elemento na lista e outra para o anterior. Além disso, cada AABB deve guardar referências para os dois elementos que a contém em cada lista, permitindo assim que,

[1] O método foi primeiramente chamado de *sort and sweep* na tese de Baraff (1992), sendo posteriormente popularizado por Cohen et al. (1995) com o nome de *sweep and prune* (SAP).

dado uma AABB, seja possível acessar de forma rápida a posição das projeções de seus extremos em cada lista.

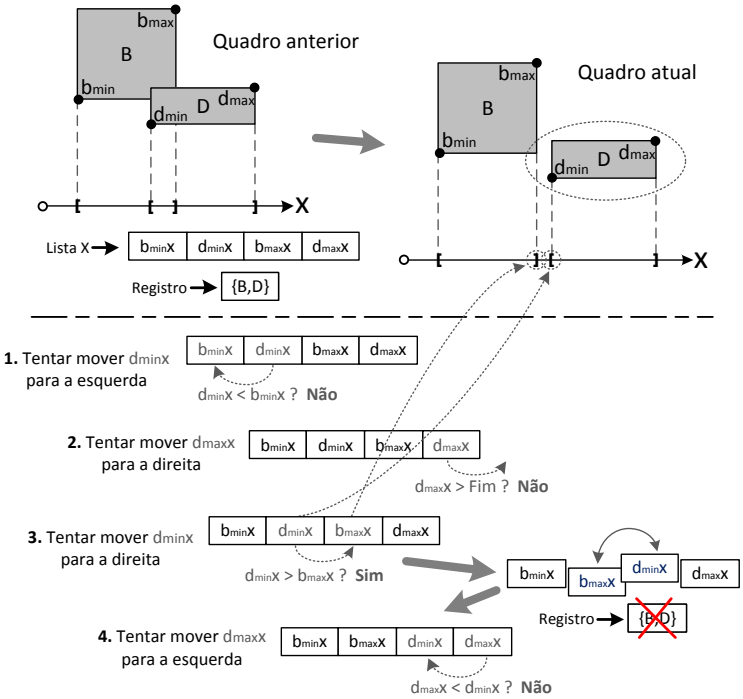

Figura 12: Ilustração da aplicação do SAP na lista associada ao eixo x após o objeto D se mover, tornando desatualizados a lista e o registro de pares. O passo 3 verifica que a lista está desordenada, aplicando então as medidas necessárias para reordenar a lista e atualizar o registro.

As listas estão ordenadas em ordem crescente com relação ao valor das projeções das AABBs. Por exemplo, no instante específico ilustrado pela Figura 11, o conteúdo de cada lista é

$$Lista\ X = (a_{min}x,\ a_{max}x,\ c_{min}x,\ b_{min}x,\ c_{max}x,\ d_{min}x,\ b_{max}x,\ d_{max}x),$$

$$Lista\ Y = (a_{min}y,\ c_{min}y,\ c_{max}y,\ a_{max}y,\ d_{min}y,\ b_{min}y,\ d_{max}y,\ b_{max}y),$$

onde $a_{min}x$ é a projeção do valor *mínimo* de A no eixo X.

A Figura 12 ilustra o funcionamento do SAP: supõe-se que, no instante inicial, já esteja registrada a informação de que o par $\{B, D\}$ está se intersectando. No instante seguinte, apenas D deslocou-se um pouco, e somente para a direita, de forma que agora B e D não mais se intersectam. É necessário, portanto, excluir $\{B, D\}$ do registro e atualizar a lista X, que agora se encontra não-ordenada, pois $d_{min}x > b_{max}x$ mas $d_{min}x$ está antes de $b_{max}x$. Dado que cada AABB tem referências para os elementos da lista nos quais seus extremos estão inseridos, e que cada elemento tem referências para seus vizinhos, é possível rapidamente verificar a partir de D (a AABB que se moveu) que seu extremo $d_{min}x$ deve ser movido para a direita na lista X. Além disso, o fato de que o menor valor da projeção de D no eixo X ($d_{min}x$) passou para uma posição na lista que está mais à direita do maior valor da projeção de B no mesmo eixo ($b_{max}x$), significa que o par $\{B, D\}$ não está mais se intersectando, portando, deve ser removido do registro.

Genericamente, quando um objeto qualquer se move, deve-se verificar se os extremos desse objeto ainda estão nas posições corretas em cada lista. Uma descrição completa e genérica do algoritmo é fornecida a seguir.

Seja E a lista associada a um eixo e qualquer (x, y ou z), e $E[i]$ o elemento na i-ésima posição dessa lista. A lista E encontra-se inicialmente ordenada. Para cada AABB K que se move, deve-se aplicar a seguinte sequência de passos para reordenar as listas e atualizar o registro de pares:

1. *Tentar mover $k_{min}e$ para a esquerda* – Parte-se da posição atual i de $k_{min}e$ e define-se inicialmente $j = i$. Enquanto $k_{min}e < E[j-1]$, deve-se decrementar o valor de j até no máximo $j = 2$ (segundo elemento da lista E) e testar, para cada j, se $E[j-1]$ é um valor de máximo. Caso seja, significa que as projeções de K e da AABB referente à projeção $E[j-1]$ se intersectam no eixo e, existindo então a possibilidade de elas se intersectarem em todos os eixos. Essa possibilidade é verificada realizando-se um teste completo de intersecção entre essas AABBs. Se o teste retornar que existe a intersecção, deve-se adicionar o par referente a tais AABBs ao registro de pares, caso ainda não tenha sido

adicionado. Finalmente, quando $k_{min}e < E[j-1]$ não for verdadeiro ou j for menor do que 2, se $j \neq i$ (ou seja, conseguiu-se constatar que $k_{min}e$ estava fora de ordem na lista E), $k_{min}e$ deve ser removido da posição i e inserido na posição entre j e $j+1$.

2. *Tentar mover $k_{max}e$ para a direita* – Análogo ao passo 1.

3. *Tentar mover $k_{min}e$ para a direita* – Parte-se da posição atual i de $k_{min}e$ e define-se inicialmente $j = i$. Enquanto $k_{min}e > E[j+1]$, deve-se decrementar o valor de j até no máximo $j = n-1$ (sendo n o tamanho da lista E) e testar, para cada j, se $E[j+1]$ é um valor de máximo. Caso seja, deve-se remover do registro o par formado por K e pela AABB referente à projeção $E[j+1]$. Finalmente, quando $k_{min}e > E[j+1]$ não for verdadeiro ou j for maior do que $n-1$, se $j \neq i$ (ou seja, conseguiu-se constatar que $k_{min}e$ estava fora de ordem na lista E), $k_{min}e$ deve ser removido da posição i e inserido na posição entre $j-1$ e j.

4. *Tentar mover $k_{max}e$ para a esquerda* – Análogo ao passo 3.

Ericson (2005) apresenta uma implementação eficiente desse algoritmo, além de uma alternativa usando *arrays* ao invés de listas encadeadas.

Como em uma simulação física os objetos geralmente se movem pouco entre quadros consecutivos, o trabalho realizado pelo SAP a cada novo quadro para atualizar as listas e o registro é pequeno, pois as listas tendem a permanecer ordenadas ou quase ordenadas.

4.2 FASE RESTRITA

Os pares de objetos em potencial colisão encontrados na fase ampla chegam a fase restrita, onde será realizado um teste de colisão mais minucioso.

O algoritmo para realizar o teste na fase restrita varia de acordo com o tipo e a geometria dos objetos presentes em cada par. Por exemplo, para testar a colisão entre dois objetos rígidos convexos, o simulador desenvolvido neste trabalho utiliza uma técnica baseada no algoritmo *GJK* (GILBERT; JOHNSON; KEERTHI, 1988; CAMERON,

1997; BERGEN, 2003). Uma descrição detalhada desse algoritmo é fornecida por Souza (2011). Já para testar a colisão de um tecido (objeto deformável) com um objeto estático (e.g., manequim), utiliza-se a técnica de "campos de distância", a qual é descrita a seguir (subseção 4.2.1).

A escolha do algoritmo correto com base no tipo dos objetos envolvidos é realizada utilizando-se um mecanismo de *double dispatch* (NOBREGA; CARVALHO; WANGENHEIM, 2008).

4.2.1 Campos de Distância

"Campos de distância" (*distance fields*) é o nome da estrutura usada pelo simulador desenvolvido neste trabalho para realizar detecção de colisão de tecidos com objetos estáticos de maneira eficiente. Essa técnica é descrita em diversos trabalhos (FRISKEN et al., 2000; FUHRMANN; SOBOTKA; GROSS, 2003; TESCHNER et al., 2005).

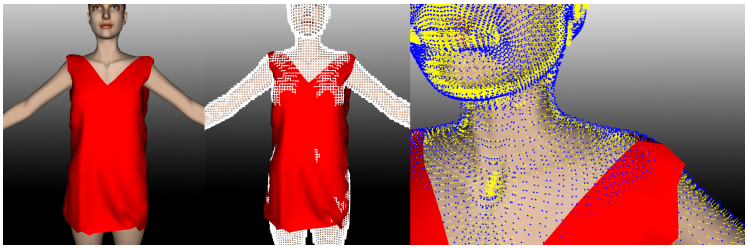

Figura 13: Campos de distância para um modelo tridimensional de manequim. A imagem central mostra as células não vazias da grade inicialmente gerada. Na imagem da direita, os pontos em amarelo estão colidindo com o corpo do manequim; as linhas de mesma cor são os vetores normais nesses pontos. Os pontos em azul têm valor de distância maior do que zero, portanto, não estão colidindo com o manequim.

A ideia é gerar uma grade uniforme (subseção 4.1.2) que englobe a malha de triângulos do modelo estático. Para cada vértice da grade (cada célula é formada por oito vértices), é calculada e armazenada a menor distância desse ponto até a malha de triângulos. Vértices

internos à malha contêm um valor de distância negativo. Para saber se uma partícula do tecido intersecta a malha estática, verifica-se em qual célula da grade tal partícula está, e a distância dessa partícula à malha é aproximada via interpolação trilinear das distâncias que foram previamente calculadas para os oito vértices dessa célula. Distâncias com valor negativo ou igual a zero indicam que o ponto está na região interna ou na superfície da malha.

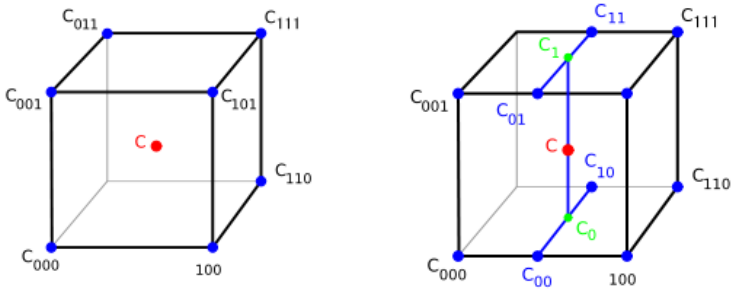

Figura 14: À esquerda, os vértices da AABB que contém o ponto C. À direita, os pontos intermediários usados na interpolação trilinear no ponto C dos valores de distância pré-calculados para cada vértice da AABB.

Seja $C \in \mathbb{R}^3$ o ponto que se deseja calcular a distância para a malha estática. C está no interior de uma AABB (ver subseção 4.1.1) que representa uma célula da grade uniforme que contém a malha. Seja C_{ijk}, $i,j,k \in \{0,1\}$, um dos oito vértice dessa AABB, onde o ponto mínimo é representado pelo vértice C_{000}, e o ponto máximo por C_{111}, conforme ilustrado na imagem esquerda da Figura 14.

Sejam x, y e z os valores das coordenadas do ponto C. Tem-se que x_d, y_d e z_d são os valores dessas coordenadas expressas em relação ao ponto mínimo da AABB da célula da grade, C_{000}, tal que

$$x_d = (x - x_0)/(x_1 - x_0),$$

$$y_d = (y - y_0)/(y_1 - y_0),$$

$$z_d = (z - z_0)/(z_1 - z_0),$$

onde x_0 é coordenada do vértice C_{000} e x_1 coordenada de C_{111}, o que é definido de forma similar para y_0, y_1, z_0 e z_1.

Com isso, seguindo a convenção adotada na imagem à direita na Figura 14, a interpolação pode ser feita primeiramente no eixo x:

$$C_{00} = V_{000}(1 - x_d) + V_{100}x_d,$$

$$C_{10} = V_{010}(1 - x_d) + V_{110}x_d,$$

$$C_{01} = V_{001}(1 - x_d) + V_{101}x_d,$$

$$C_{11} = V_{011}(1 - x_d) + V_{111}x_d,$$

onde V_{ijk} indica o valor da distância pré-calculada para o vértice C_{ijk}. Em seguida, interpola-se no eixo y:

$$C_0 = C_{00}(1 - y_d) + C_{10}y_d,$$

$$C_1 = C_{01}(1 - y_d) + C_{11}y_d,$$

E, finalmente, interpola-se no eixo z para obter o valor aproximado da distância no ponto C:

$$distância = C_0(1 - z_d) + C_1 z_d.$$

No caso de haver uma intersecção (i.e., se a distância calculada for menor ou igual a zero), necessita-se também calcular o vetor normal à superfície de colisão, pois ele será necessário para qualquer algoritmo de tratamento de colisão utilizado (BRIDSON; FEDKIW; ANDERSON, 2002). Esse vetor pode ser facilmente calculado utilizando-se os campos de distância. Segundo Frisken et al. (2000), para obtê-lo, basta normalizar o vetor gradiente dos campos de distância na superfície.

Define-se a função $h(x,y,z) : \mathbb{R}^3 \to \mathbb{R}$, , que aceita um ponto qualquer como entrada e retorna a distância desse ponto até a malha. Ou seja, h implementa a interpolação trilinear descrita anteriormente.

Seja $(a,b,c) \in \mathbb{R}^3$ o vetor gradiente no ponto (x,y,z) quando $h(x,y,z) \leq 0$. Usando a mesma notação anterior, tem-se que

$$a = h(x_1, y_d, z_d) - h(x_0, y_d, z_d),$$

$$b = h(x_d, y_1, z_d) - h(x_d, y_0, z_d),$$
$$c = h(x_d, y_d, z_1) - h(x_d, y_d, z_0).$$

Então, o vetor normal à superfície da malha estática no ponto (x, y, z) quando $h(x, y, z) \leq 0$ é simplesmente o vetor (a, b, c) normalizado.

Na técnica de campos de distância descrita aqui, a etapa de maior custo computacional é a realizada inicialmente: o cálculo das distâncias até a malha para cada vértice da grade. No simulador desenvolvido neste trabalho, esta etapa é realizada *offline*; isto é, para um determinado modelo estático, a grade e as distâncias são previamente geradas e salvas em disco. Assim, quando necessário, essas informações são rapidamente carregadas.

5 RASGO, CORTE E COSTURA

A simulação de rasgo de tecidos é, geralmente, um processo não-trivial, uma vez que requer a capacidade de atualizar dinamicamente as diferentes representações e estruturas de dados usadas na simulação (ver seção 5.1). E, para simulações interativas, esse processo deve ser realizado da maneira mais eficiente possível.

Este capítulo mostra uma ideia simples que pode ser usada para simular rasgo e corte de tecidos (seção 5.2). Apesar de não prover resultados fisicamente precisos, é uma técnica eficiente, o que a torna ideal para aplicações interativas. A ideia foi formalmente mencionada no trabalho de Müller et al. (2007); contudo, já havia sido discutida pela comunidade de pesquisa de maneira informal. Como contribuição original, este capítulo sugere uma otimização para essa técnica utilizando uma estrutura de dados *half-edge* especialmente adaptada.

Outra questão a ser considerada é como costurar tecidos. Isso é importante porque, no mundo real, peças de roupa são, geralmente, constituídas por diferentes pedaços planos de tecidos – representados graficamente por polígonos, chamados de *moldes* (ver Figura 15) – que são unidos por costuras (Figura 24). A seção 5.3 mostra como esse processo é contemplado na simulação. Antes, porém, a seção 5.1 apresenta como o tecido é representado internamente pelo simulador e como essa representação é gerada com base nos moldes de cada peça de roupa.

5.1 REPRESENTAÇÃO GEOMÉTRICA

Sistemas de simulação de tecidos precisam lidar com, pelo menos, duas representações internas. A primeira, chamada aqui de *representação física*, é mais adequada ao processo de simulação – formada por partículas e restrições de movimento, cujos detalhes variam de acordo com o modelo matemático escolhido (ver capítulo 3). A segunda, chamada de *representação geométrica* (ou gráfica), visa facilitar a renderização dos objetos na tela. Uma vez que as placas gráficas são máquinas projetadas para desenhar triângulos de forma eficiente,

uma *malha de triângulos* é o formato mais comumente usado para essa representação. A cada quadro da simulação, a representação geométrica precisa ser atualizada de acordo com a representação física.

Acredita-se que o usuário de um simulador não deve conhecer detalhes dos modelos e representações usadas internamente. Por isso, para definir a forma inicial para cada pedaço de tecido, um simulador moderno deve requerer do usuário apenas polígonos (conjunto planar de pontos), que representam os moldes de cada peça. O simulador precisa, então, gerar uma malha de triângulos a partir de cada polígono fornecido.

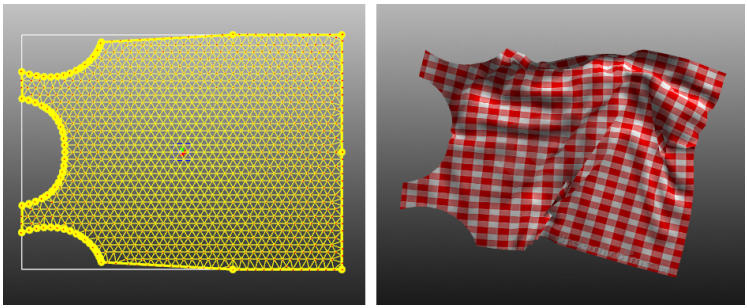

Figura 15: À esquerda, molde (polígono) de uma peça de roupa. À direta, tecido que foi gerado a partir desse molde sendo simulado.

5.1.1 Geração de Malha de Triângulos

A geração de uma malha de triângulos para um polígono qualquer (i.e., côncavo, convexo, com buracos) de maneira eficiente é um problema complexo de geometria computacional, porém, bem tratado na literatura (KIRKPATRICK; KLAWE; TARJAN, 1992; RUPPERT, 1995; AMATO; GOODRICH; RAMOS, 2000). No simulador desenvolvido, optou-se pelo uso da biblioteca *Triangle* (SHEWCHUK, 1996) para realizar essa tarefa.

A *Triangle* aceita um conjunto de pontos e segmentos (um *planar straight-line graph*, PSLG), que representam as arestas de um polígono. Ela retorna um novo conjunto de pontos – conjunto possivel-

mente maior e que contém os pontos originais – e um conjunto de índices que indicam como esses pontos são ligados para formar os triângulos gerados (ver Figura 16). Os pontos adicionais incluídos pela *Triangle* são chamados de *Steiner points*. Esses pontos podem ser gerados onde for necessário para a obtenção de uma boa malha (i.e., sem triângulos degenerados). Contudo, visando facilitar a costura dos moldes, caso existam *linhas de costura* especificadas (maiores detalhes na seção 5.3), alguns desses pontos são criados manualmente antes de ser realizada a chamada à biblioteca.

A Figura 17 mostra um exemplo de malha que foi gerada pela *Triangle* e simulada no sistema desenvolvido neste trabalho.

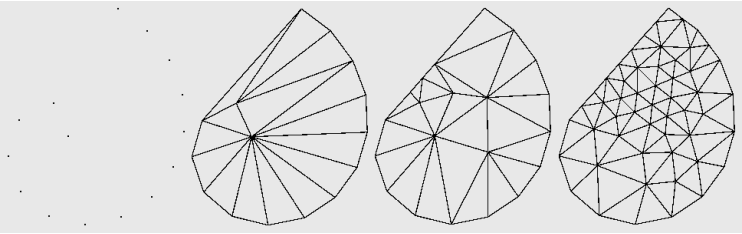

Figura 16: Exemplo de triangularização realizada pela biblioteca *Triangle* (SHEWCHUK, 1996).

5.1.2 Representação *Half-Edge*

No simulador desenvolvido, uma malha de triângulos é representada por uma estrutura simples, composta por um vetor de vértices (pontos) e um vetor de faces. Cada face é formada por três valores, inteiros positivos, que são os índices dos vértices que constituem a face. Essa estrutura permite rápida transferência dos dados para a placa gráfica. Contudo, algumas operações não são realizadas de forma eficiente nessa representação, como por exemplo, consultas de adjacência (e.g., determinar quais são as faces adjacentes a uma face qualquer, ou saber quais arestas compartilham um certo vértice). Para otimizar tarefas desse tipo, o simulador utiliza internamente uma representação chamada de *half-edge*.

A estrutura de dados *half-edge* pode ser usada para representar determinadas malhas de polígonos. Ela permite realizar várias consultas de adjacência em tempo constante. Além disso, é possível implementar essa representação de forma que os tamanhos de suas estruturas de dados internas permaneçam sempre os mesmos; com isso, nenhum alocamento dinâmico de memória é necessário (listas encadeadas ou *arrays* tradicionais são desnecessários). Tais propriedades fazem da *half-edge* uma boa opção para os propósitos deste trabalho. Uma descrição dessa estrutura é fornecida por DeCoro e Pajarola (2002).

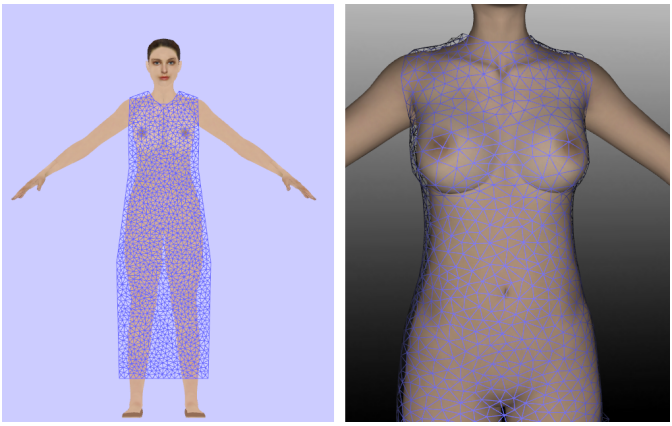

Figura 17: Geração de malha de triângulos. Detalhe da malha gerada para um molde (polígono) de parte da peça de roupa de um manequim. A imagem mostra a malha antes (à esquerda) e depois de ser simulada.

Um ponto a ser considerado é que a *half-edge* é capaz de representar corretamente apenas malhas classificadas como uma variedade de dimensão dois (*2-manifold*). Para uma malha de triângulos, isso significa que não é possível representar, por exemplo, malhas onde uma aresta é compartilhada por mais de duas faces, ou malhas onde duas faces compartilham um vértice sem compartilhar também uma aresta (ver Figura 21). Maiores considerações sobre isso são feitas na subseção 5.2.2.

5.2 DIVISÃO DE VÉRTICE

Divisão de vértice é o nome dado ao algoritmo básico usado para implementar rasgo e costura no simulador desenvolvido. A Figura 18 ilustra o seu funcionamento. Nesse exemplo, o vértice central de uma malha de triângulos foi escolhido para ser dividido por um plano arbitrário, chamado de plano de divisão. Um novo vértice é criado no mesmo local do vértice escolhido (na figura, ele aparece em uma posição diferente apenas para fins de ilustração). A representação *half-edge* da malha é usada aqui para se obter eficientemente todos os triângulos conectados ao vértice dividido. Então, o plano é usado para classificar esses triângulos em dois grupos: os que estão acima e os que estão abaixo do plano. Os triângulos que estão abaixo do plano passam a apontar para o novo vértice. No exemplo da figura, esses são os triângulos numerados com 4, 5 e 6. Os triângulos 1, 2 e 3 permanecem inalterados.

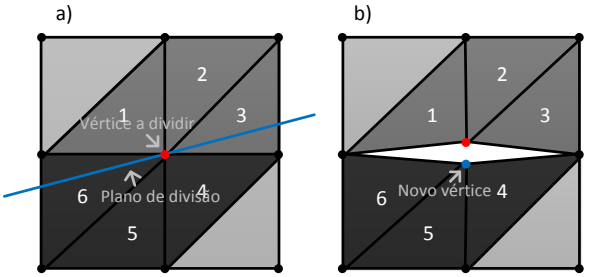

Figura 18: Algoritmo de divisão de vértice. O vértice central (em vermelho) é escolhido para ser dividido por um plano arbitrário (a) e, então, um novo vértice é criado (b). O novo vértice é atribuído aos triângulos abaixo do plano de divisão.

Uma nova partícula precisa ser criada na representação física (capítulo 3). A massa dessa nova partícula é definida como sendo metade da massa original da partícula associada ao vértice que foi dividido, que por sua vez, também tem sua massa reduzida à metade.

A outra parte do algoritmo é atualizar as restrições internas de movimento, o que depende da representação física usada. Basi-

camente, é necessário inserir novas restrições de distância (*stretch*) para segurar a nova partícula e remover totalmente, caso existam, as restrições de dobra (*bend*) relacionadas ao vértice que foi dividido. Ainda, qualquer restrição relacionada aos triângulos abaixo do plano deve apontar para a nova partícula. A representação *half-edge* pode ser usada para acelerar a atualização de restrições também.

Deve-se notar que nem todos os vértices podem ser divididos: é preciso existir ao menos um triângulo em cada lado do plano de divisão. Caso isso não ocorra, a rotina que implementa a divisão de vértice não deve prosseguir.

5.2.1 Rasgo

Com a rotina de divisão de vértice, o rasgo do tecido é implementado com um processo simples. Sempre que a tensão de uma aresta da malha (ou uma restrição de distância entre duas partículas) atingir um determinado limiar, seleciona-se um dos vértices da aresta para ser dividido. Se ambos estão aptos ao processo, seleciona-se o vértice associado à partícula com o maior valor de massa.

Se a aresta tensionada em demasia é formada pelos vértices A e B, e A é o vértice que foi escolhido para ser dividido, o plano de divisão é dado pelo ponto A e o vetor normal ao plano tem a direção de $B - A$.

Figura 19: Sequência de quadros de um protótipo de simulação interativa onde um pedaço de tecido é rasgado pelo usuário, que "segura" um vértice da malha e o arrasta para baixo.

Observa-se que, quando um dos vértices de uma aresta tensionada é dividido, a nova configuração de restrições na representação

física irá impor a velocidade necessária para separar a nova partícula, criando um pequeno buraco na malha do tecido. O resultado é ilustrado na Figura 19.

5.2.2 Modificando a *Half-Edge*

Ao executar a divisão de vértice, além de atualizar a malha de triângulos e a representação física, é necessário atualizar também a representação *half-edge*. Conforme mencionado na subseção 5.1.2, essa estrutura só é capaz de representar corretamente malhas classificadas como uma variedade de dimensão dois (*2-manifold*). Na figura 21, duas faces compartilham um único vértice e não compartilham nenhuma aresta, o que é um exemplo de uma malha de triângulos que não atende tal requisito. Infelizmente, após realizar algumas execuções da rotina de divisão de vértices da forma que ela foi descrita, é possível deparar-se com uma malha exatamente nesse estado.

A solução encontrada para essa questão – a qual viabilizou utilizar malhas até mesmo nesse estado degenerado – foi realizar uma pequena modificação na implementação padrão da estrutura de dados *half-edge*.

Na implementação padrão, uma malha de triângulos é representada por um conjunto de vértices, um conjunto de faces (triângulos) e um conjunto de "meias-arestas" (*half-edges*). Cada meia-aresta deve apontar para quatro dados: uma face, um vértice dessa face, uma próxima meia-aresta nessa face e uma meia-aresta "par", que é uma meia-aresta em uma face adjacente (ver DeCoro e Pajarola (2002) para maiores detalhes). Na modificação proposta, adiciona-se um quinto atributo a uma meia-aresta, chamado de "par fantasma" (ver Figura 20). Inicialmente, esse atributo deve receber um valor *default* que indique um estado inválido (e.g., valor nulo ou negativo).

A rotina que aplica a divisão de vértice na representação *half-edge* recebe dois conjuntos contendo os índices dos triângulos em cada lado do plano de divisão (ver Figura 18). A rotina deve desassociar estes dois conjuntos apagando os valores dos atributos de par nas meias-arestas que formam as arestas que separam esses dois conjuntos. Na implementação realizada, o valor dos atributos de par é simplesmente salvo no atributo "par fantasma" antes de ser apagado (setado com um

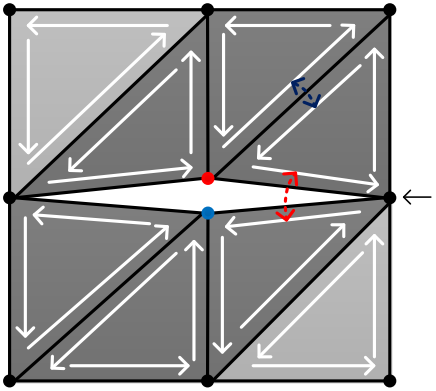

Figura 20: Após a execução da divisão de vértice. A mesma malha da Figura 18 onde o vértice central foi dividido. As setas brancas mostram as "meias-arestas". A seta azul mostra um par regular de meias-arestas, enquanto que a seta em vermelho indica um par fantasma, formado após a divisão.

valor inválido). Essa é a única parte do algoritmo onde a valor do par fantasma é modificado após ser inicializado. A única rotina que precisa ler o valor do par fantasma é a que procura por faces conectadas a um vértice (chamada pela divisão de vértice). O valor de par é usado para mover para faces adjacentes, assim, essa rotina precisa apenas verificar também o par fantasma quando o par regular contiver um valor inválido. Assume-se que a malha inicial (antes de ser aplicada qualquer divisão de vértice) é uma malha de triângulos válida na representação *half-edge*, i.e., é uma variedade de dimensão dois; caso contrário, essa modificação não funcionaria da maneira esperada.

Sem o atributo do par fantasma, a consequência de se ter uma malha que não é uma variedade de dimensão dois é que alguns vértices não poderão ser divididos quando, na verdade, eles deveriam poder. Na Figura 20, por exemplo, o vértice indicado pela seta preta não poderia ser dividido por um plano horizontal porque a implementação padrão da *half-edge* não permitiria encontrar triângulos para os dois lados do plano de divisão.

Ressalta-se que o atributo de par fantasma existe apenas para

manter funcionando adequadamente a rotina que calcula as faces que rodeiam um determinado vértice. Todas as outras rotinas que implementam consultas de adjacência usando a *half-edge* utilizam o atributo de par regular, portanto, não necessitam ter sua implementação nem funcionamento alterados.

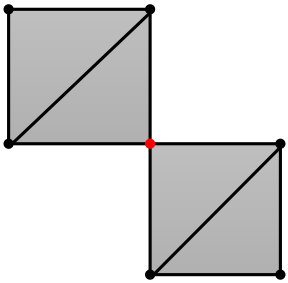

Figura 21: Exemplo de malha de triângulos que não pode ser classificada como uma variedade de dimensão dois (*2-manifold*): duas faces compartilham um único vértice (em vermelho) sem também compartilhar uma aresta.

5.2.3 Ferramenta de Corte

O algoritmo de divisão de vértice pode ser usado para implementar uma "ferramenta de corte" ou tesoura, conforme ilustrado na Figura 22. A rotina que implementa essa tarefa recebe dois raios, R_A e R_B, que são fornecidos pelo usuário quando este seleciona dois pontos na tela. Cada raio é definido por um ponto de origem O e um vetor de direção V.

Define-se, então, um plano P que passa pelo ponto O_A (ponto de origem de R_A), e um vetor normal N dado pelo produto cruzado entre o vetor $O_A - O_B$ e V_A:

$$N = (O_A - O_B) \times V_A.$$

Ainda, define-se outros dois planos, P_A e P_B, que serão usados como limites laterais. P_A e P_B passam por O_A e O_B, respectivamente,

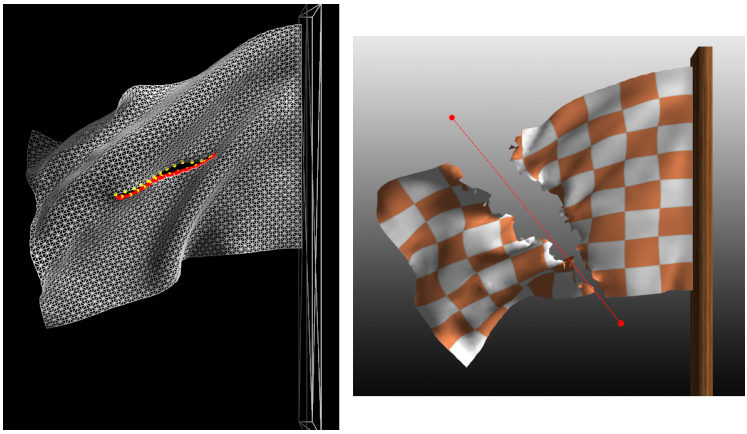

Figura 22: Usando a ferramenta de corte. À esquerda, um pequeno corte horizontal é feito na bandeira; os vértices divididos são mostrados em amarelo. À direita, o usuário escolhe dois pontos na tela para definir o segmento de reta em vermelho, que é usado para cortar a bandeira durante a simulação.

e seus vetores normais são

$$N_A = V_A \times N,$$

$$N_B = N \times V_B.$$

A rotina deve iterar sobre todos os triângulos na malha escolhida para corte a fim de obter somente os triângulos que intersectam o plano P. Para cada triângulo, a intersecção é testada separando-se seus vértices em dois conjuntos, um para cada lado (positivo e negativo) do plano P. Se o triângulo não possuir vértices em ambos os conjuntos, ele é ignorado. Arbitrariamente, decidiu-se por aplicar a divisão de vértice nos vértices que estão no conjunto que representa o lado positivo do plano P. Cada vértice nesse conjunto deve estar também no lado positivo tanto de P_A quanto de P_B, caso contrário, o triângulo também é ignorado, pois está fora dos limites laterais. O plano de divisão usado em cada chamada a rotina de divisão de vértice é definido pela própria posição de cada vértice e por N como vetor

normal.

Na prática, como malhas regulares, tal algoritmo as vezes tem o indesejável resultado de não separar completamente a malha no segmento de reta especificado. Um resultado melhor pode ser obtido alternando-se o vetor normal do plano de divisão usado em cada chamada a rotina de divisão de vértice. Portanto, somente para os triângulos que têm dois vértices, C e D, no lado positivo do plano P, não se usa N como normal do plano de divisão. Ao invés disso, usa-se

$$N^{'} = N_{tri} \times (D - C),$$

onde N_{tri} é a normal do triângulo sob análise.

5.3 COSTURAS

Para simular costuras, cria-se restrições de movimento entre as partículas semelhantes as restrições de distância mostradas nas subseções 3.1.1 e 3.2.1. A diferença é que, nas restrições de costura, a distância que se deseja manter entre as partículas é igual a zero.

A fim de facilitar a operação de costura em malhas de triângulos – malhas estas geradas a partir dos moldes de cada peça, conforme comentado na seção 5.1 – utiliza-se uma abstração chamada de *linha de costura*. Uma linha de costura é um conjunto de vértices adjacentes em uma malha de triângulos, i.e., partindo-se de um vértice e "caminhando" pelas arestas, pode-se atingir todos os outros vértices no conjunto.

Uma linha de costura deve ser unida (ou "costurada") com outra linha de costura. Unir duas linhas de costura significa adicionar novas "restrições de costura" na representação física, fazendo com que os vértices de uma linha se juntem com os da outra.

O ideal é que duas linhas de costura que serão unidas possuam a mesma quantidade de vértices (ver Figuras 23 e 24). Quando for viável gerar (ou regerar) malhas de triângulos em função das linhas de costura e das uniões entre elas estabelecidas (conforme foi mencionado na subseção 5.1.1), isso não é um problema. Outra situação, porém, é quando se tem malhas de triângulos prontas e precisa-se gerar linhas de costura sobre elas. A seguir, apresenta-se um algoritmo

para gerar linhas de costura nessa situação.

Um polígono P é um vetor de vértices V tal que todos os vértices em V pertencem ao mesmo plano em $\mathbb{R}^3$. Uma malha de triângulos T, gerada para o polígono P, é formada por

1. T_{pontos}, um vetor de pontos, tal que todos os pontos estão dentro da região definida por P;

2. $T_{indices}$, um vetor de triplas, tal que cada tripla contém os índices de três pontos em T_{pontos}; ou seja, cada tripla representa um triângulo da malha T.

Uma linha de costura S é um subconjunto de T_{pontos}, tal que todos os pontos em S são adjacentes em T, i.e., existe um caminho pelas arestas de T que contém todos os pontos em S.

Seja $\{p_1, p_2\}$ um par de pontos tal que p_1, p_2 são as posições de dois vértices em T. Deseja-se achar um caminho pelas arestas de T, indo de p_1 a p_2, de forma que a soma das distâncias de cada ponto nesse caminho ao segmento de reta definido por $\overrightarrow{p_1 p_2}$ seja mínimo.

Inicialmente, a linha de costura S é um conjunto vazio. Os passos seguintes resumem o algoritmo.

1. Definir p_1 como o ponto atual p;

2. Inserir p em S;

3. Criar um conjunto A contendo todos vértices adjacentes a p em T (a representação *half-edge* pode ser usada aqui);

4. Chamar uma função de estimativa F que irá prover uma pontuação para cada elemento de A. Definir o elemento com maior pontuação como sendo o novo ponto p;

5. Se p é igual a p_2, parar; S contém a linha de costura desejada. Caso contrário, ir para o passo 2.

A função de estimativa usada no passo 4, $F : \mathbb{R}^3 \to \mathbb{R}$, recebe um ponto p qualquer e retorna $\frac{1}{M+N}$, onde M é a distância de p ao segmento de reta $\overrightarrow{p_1 p_2}$ e N é a distância de p a p_2, o ponto final.

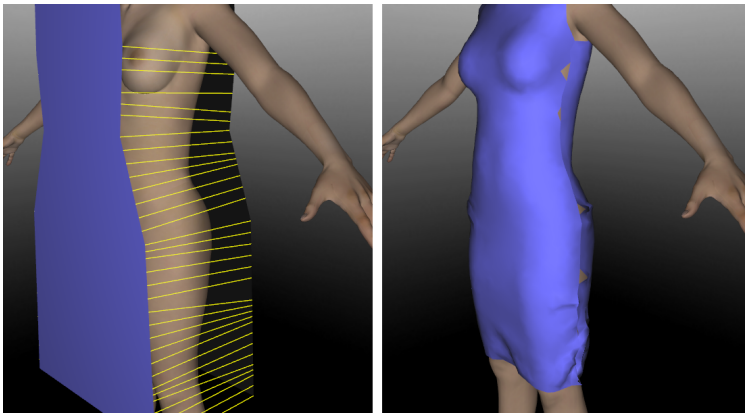

Figura 23: Exemplo de linhas de costura mal geradas. Uma quantidade diferente de vértices entre duas linhas de costuras que serão unidas acarreta em um fechamento incorreto das costuras nos moldes.

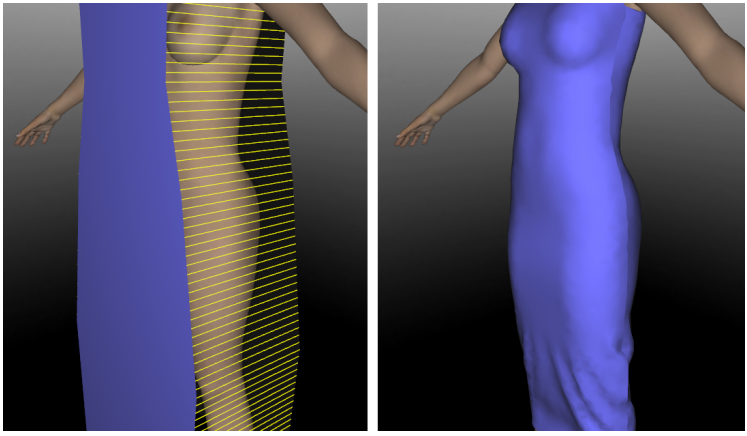

Figura 24: Exemplo de linhas de costura bem geradas. Mesmo número de vértices é gerado para duas diferentes linhas de costuras que serão unidas.

6 IMPLEMENTAÇÃO E RESULTADOS

O projeto do qual este trabalho faz parte foi iniciado com o propósito de conceber ferramentas de simulação e visualização para serem usadas em aplicações de projeto e confecção de roupas.

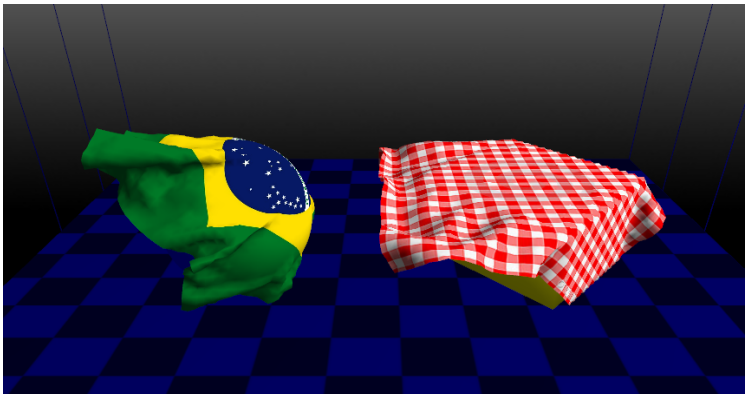

Figura 25: Detecção e tratamento de colisão entre tecidos e corpos rígidos e simulação de vento.

As técnicas de simulação de tecidos estudadas e descritas nesta dissertação foram implementadas como extensões das funcionalidades de um simulador de *corpos rígidos* desenvolvido em trabalhos anteriores (SOUZA, 2011), chamado de *Planck*.

O simulador *Planck* é apenas um dos módulos de um conjunto maior de ferramentas. A etapa de visualização não faz parte do escopo desta pesquisa. Assim, para a tarefa de renderização, utiliza-se a biblioteca de código aberto chamada *Ogre* (ogre3d.org), a qual tem a finalidade de agilizar o desenvolvimento de aplicações gráficas fornecendo abstrações de alto nível. Foi desenvolvido também um módulo chamado de *Plangre*, o qual provê abstrações que facilitam a integração da *Planck* (simulação) com a *Ogre* (renderização).

A linguagem de programação utilizada em todo o projeto é *C++*. O código utiliza somente soluções multiplataforma. *CMake* (cmake.org) é usado para auxiliar o processo de compilação. Os com-

piladores *Visual C++* e *MinGW* (mingw.org) são suportados em sistemas *Windows* e o *GCC* em *GNU/Linux*.

Figura 26: Pedaço de tecido é preso nos ombros de um manequim virtual para simular uma capa de super-herói.

Ao longo do processo de desenvolvimento foram criados protótipos para testar as técnicas estudadas. Vídeos que mostram a execução do simulador e alguns desses protótipos podem ser encontrados em:

- http://youtu.be/hn4v2EzXjlo
- http://youtu.be/61ky6EkCy6c

A Figura 25 ilustra uma cena onde dois pedaços de tecido, inicialmente suspensos, caem sobre dois corpos rígidos estáticos, um com o formato de uma esfera e, o outro, de um paralelepípedo. Uma força que simula a ação do vento é aplicada em cada triângulo das malhas que formam os tecidos.

A Figura 26 mostra uma cena onde um pedaço de tecido – no formato de uma capa de super-herói – tem algumas de suas partículas fixadas nas costas de um manequim. Para fixar uma partícula, basta definir sua massa como tendo valor igual a zero. Esse valor é tratado pelo simulador como um valor de “massa infinita”, fazendo com que a partícula não possa ser deslocada.

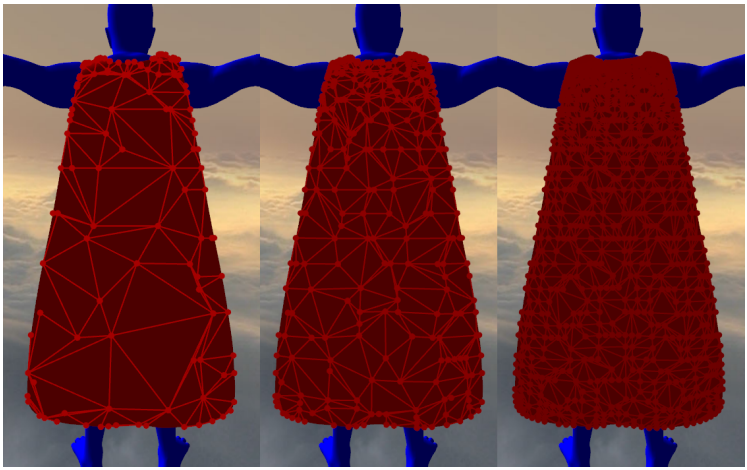

Figura 27: Teste da técnica sugerida por Müller (2008), onde as restrições de distância são aplicadas em níveis de hierarquia. A imagem mostra as restrições criadas para três desses níveis.

Figura 28: Simulação de uma peça de roupa sobre o corpo de um manequim.

A Figura 27 mostra um protótipo que foi desenvolvido para testar a implementação de uma técnica sugerida por Müller (2008) que é complementar ao *PBD* (seção 3.2). Essa técnica não é abordada nesta dissertação.

A Figura 28 ilustra uma aplicação desenvolvida para testar o caimento de uma roupa simulada sobre o corpo de um manequim virtual. Os moldes bidimensionais (ver seção 5.1) especificados pelo usuário são costurados e simulados no corpo do manequim em um ambiente tridimensional.

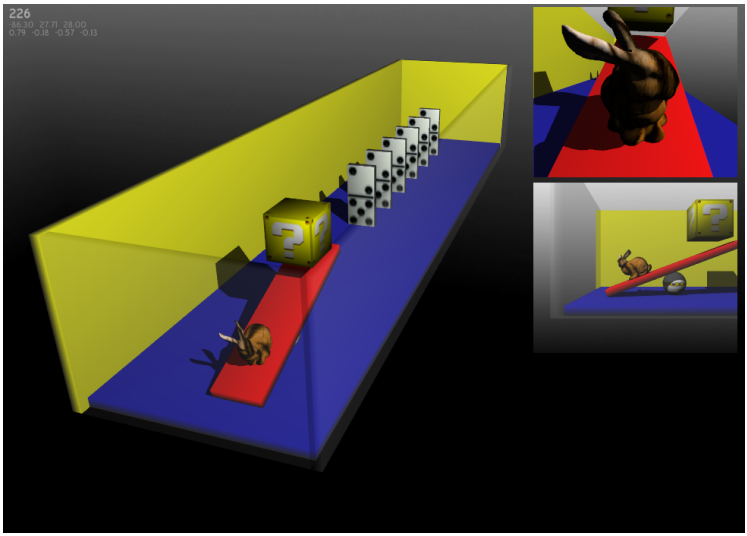

Figura 29: Exemplo de simulação de corpos rígidos na *Planck*.

A Figura 29 mostra uma cena que foi implementada para testar a simulação de corpos rígidos da *Planck*. Um objeto em forma de coelho é lançado em cima de uma fileira de dominós, derrubando-os. Uma extensa explicação sobre a simulação de corpos rígidos na *Planck* é fornecida por Souza (2011).

Protótipos também foram criados para testar as técnicas de rasgo, corte e costura descritas no capítulo 5. As Figuras 19 e 22 ilustram dois desses protótipos.

7 CONCLUSÃO

Simular o comportamento de tecidos é uma tarefa complexa, na qual precisa-se levar em consideração uma grande quantidade de parâmetros. No caso de roupas, cada peça interage fortemente com o corpo que as veste, assim como com as demais peças. Tal interação precisa ser detectada e considerada na simulação. Contudo, independente dos desafios técnicos, existe sempre a necessidade de se obter resultados mais rápidos que viabilizem a simulação interativa de indumentárias cada vez mais detalhadas.

Conforme foi colocado nas seções 1.1 e 2.2, o estado da arte em simulação de tecidos na área de animação baseada em física não provê resultados satisfatórios para certas finalidades. Com o intuito de melhorar esse aspecto, este trabalho fez contribuições pontuais, as quais são resumidas na seção 7.1.

Assim, entende-se que este trabalho atingiu seu objetivo geral, descrito na seção 1.2. Os objetivos específicos 1 e 2 são relativos ao conteúdo dos capítulos 3 e 4, respectivamente, enquanto que os objetivos 3 e 4 relacionam-se ao capítulo 5. Por fim, o capítulo 6 mostra como o objetivo 5 foi contemplado.

7.1 CONTRIBUIÇÕES

Esta dissertação aborda técnicas relativas a diferentes etapas e aspectos do processo de simulação. Contudo, destaca-se como contribuição original a otimização da técnica de divisão de vértice (usada na simulação de rasgo de tecidos), proposta na seção 5.2. Ainda, na seção 5.3 é apresentado um novo algoritmo para a geração de linhas de costura. Essa tarefa é semelhante ao *problema do caminho mínimo*, que é um problema clássico em *teoria dos grafos*. Entretanto, o cenário encontrado aqui é mais restrito, de forma que o algoritmo proposto é mais simples e intuitivo para ser usado neste contexto.

As técnicas descritas nesta dissertação foram incorporadas no simulador desenvolvido (ver capítulo 6). O simulador, como ferramenta, figura como contribuição secundária.

7.2 LIMITAÇÕES E TRABALHOS FUTUROS

Detecção de colisão por si só é um amplo campo de estudo. Este trabalho foca especificamente nas técnicas que foram empregadas para realizar detecção de colisão entre tecidos e corpos rígidos (capítulo 4). Técnicas para detecção de auto colisão – ou *self-collision*, na literatura em inglês, que é a colisão do tecido com ele mesmo – não são contempladas. Considerando os propósitos deste trabalho, a abordagem apresentada por Baraff, Witkin e Kass (2003) se mostra um das mais interessantes para tratar essa questão.

Outra questão diz respeito aos métodos de integração (capítulo 3). Dentre as diferentes abordagens presentes na literatura, os métodos que foram implementados representam dois extremos na área de animação baseada em física. O primeiro é um método clássico com foco em realismo (seção 3.1); o outro, mais recente, é focado em desempenho e facilidade de implementação (seção 3.2). Ainda assim, ambos são métodos de simulação de propósito geral, i.e., aplicam-se na simulação de qualquer tipo de tecido em, teoricamente, qualquer situação. Sabe-se que é possível combinar e adaptar diferentes técnicas (BRIDSON; MARINO; FEDKIW, 2003), ou usar simplificações visando otimizar casos específicos, como na simulação de roupas no corpo de um manequim (CORDIER; MAGNENAT-THALMANN, 2002; CORDIER, 2004). Contudo, este trabalho também não contempla tais abordagens, mas as considera como trabalhos futuros.

Há diferentes rumos que esta pesquisa pode tomar e diversos trabalhos cujos resultados podem ser incorporados ao simulador desenvolvido. A seguir, destaca-se alguns assuntos e trabalhos que o autor considera de maior relevância.

- Desempenho e estabilidade:

Com relação ao movimento do tecido gerado pela simulação, as técnicas apresentadas por Choi e Ko (2005b) e as considerações feitas por Ascher e Boxerman (2003) são as primeiras opções para melhorar o desempenho e a estabilidade do método apresentado na seção 3.1.

- Paralelização:

Alguns aspectos relativos à qualidade da simulação podem ser melhorados utilizando-se malhas de maior resolução (i.e., com maior nú-

mero de partículas) para representar cada pedaço de tecido simulado. Contudo, mais partículas acarreta em um maior custo computacional. As técnicas de paralelização mencionadas na seção 2.2 visam permitir esse aumento de resolução sem perda de desempenho.

- Costura automática de moldes:

Com relação ao problema das costuras abordado na seção 5.3, Berthouzoz et al. (2013) apresentam um método para a geração automatizada das linhas de costura com base em documentos que especificam "padrões de costura" para um conjunto de moldes. Essa abordagem é especialmente interessante porque a geração manual de costuras é uma das tarefas mais complexas dos softwares CAD para projeto de roupas (Figura 5), e que afasta muitos usuários.

- Parametrização automática:

Dado um modelo de simulação (e.g., seção 3.1 ou seção 3.2), um problema é encontrar os valores dos parâmetros nesse modelo que fazem com que um tecido simulado se comporte como um tecido feito de um determinado tipo de material (e.g., algodão, seda, etc). Bhat et al. (2003) propõem uma maneira de estimar esses parâmetros de forma automática. Um vídeo de um tecido real em movimento é comparado com o resultado de uma simulação. Algumas métricas são definidas e técnicas de visão computacional são aplicadas para extrair as informações necessárias para alterar os parâmetros da simulação buscando aproximar o movimento do tecido simulado ao do tecido real.

REFERÊNCIAS BIBLIOGRÁFICAS

AMATO, N. M.; GOODRICH, M. T.; RAMOS, E. a. Linear-time triangulation of a simple polygon made easier via randomization. *Proceedings of the sixteenth annual symposium on Computational geometry - SCG '00*, ACM Press, New York, New York, USA, p. 201–212, 2000. Disponível em: <http://portal.acm.org/citation.cfm?doid=336154.336206>.

ASCHER, U.; BOXERMAN, E. On the modified conjugate gradient method in cloth simulation. *The Visual Computer*, p. 526–531, 2003. Disponível em: <http://link.springer.com/article/10.1007/s00371-003-0220-4>.

BARAFF, D. *Dynamic simulation of nonpenetrating rigid bodies*. Tese (Doutorado), Ithaca, NY, USA, 1992. UMI Order No. GAX92-36100.

BARAFF, D.; WITKIN, A. Large steps in cloth simulation. In: *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*. New York, NY, USA: ACM, 1998. (SIGGRAPH '98), p. 43–54. ISBN 0-89791-999-8. Disponível em: <http://doi.acm.org/10.1145/280814.280821>.

BARAFF, D.; WITKIN, A.; KASS, M. Untangling cloth. In: *ACM SIGGRAPH 2003 Papers*. New York, NY, USA: ACM, 2003. (SIGGRAPH '03), p. 862–870. ISBN 1-58113-709-5. Disponível em: <http://doi.acm.org/10.1145/1201775.882357>.

BENDER, J.; WEBER, D.; DIZIOL, R. Fast and stable cloth simulation based on multi-resolution shape matching. *Computers & Graphics*, v. 37, n. 8, p. 945 – 954, 2013. ISSN 0097-8493. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0097849313001283>.

BERGEN, G. van den. *Collision Detection in Interactive 3D Environments (The Morgan Kaufmann Series in Interactive 3D Technology)*. Morgan Kaufmann, 2003. Hardcover. ISBN 155860801X. Disponível

em: <http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/155860801X>.

BERTHOUZOZ, F. et al. Parsing sewing patterns into 3d garments. *ACM Trans. Graph.*, ACM, New York, NY, USA, v. 32, n. 4, p. 85:1–85:12, jul. 2013. ISSN 0730-0301. Disponível em: <http://doi.acm.org/10.1145/2461912.2461975>.

BHAT, K. S. et al. Estimating cloth simulation parameters from video. In: *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2003. (SCA '03), p. 37–51. ISBN 1-58113-659-5. Disponível em: <http://dl.acm.org/citation.cfm?id=846276.846282>.

BRIDSON, R.; FEDKIW, R.; ANDERSON, J. Robust treatment of collisions, contact and friction for cloth animation. In: *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*. New York, NY, USA: ACM, 2002. (SIGGRAPH '02), p. 594–603. ISBN 1-58113-521-1. Disponível em: <http://doi.acm.org/10.1145/566570.566623>.

BRIDSON, R.; MARINO, S.; FEDKIW, R. Simulation of clothing with folds and wrinkles. In: *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2003. (SCA '03), p. 28–36. ISBN 1-58113-659-5. Disponível em: <http://dl.acm.org/citation.cfm?id=846276.846281>.

Enhancing GJK: computing minimum and penetration distances between convex polyhedra, v. 4. 3112–3117 vol.4 p. Disponível em: <http://dx.doi.org/10.1109/ROBOT.1997.606761>.

CARIGNAN, M. et al. Dressing animated synthetic actors with complex deformable clothes. *SIGGRAPH Comput. Graph.*, ACM, New York, NY, USA, v. 26, n. 2, p. 99–104, jul. 1992. ISSN 0097-8930. Disponível em: <http://doi.acm.org/10.1145/142920.134017>.

CHOI, K.-J.; KO, H.-S. Research problems in clothing simulation. *Comput. Aided Des.*, Butterworth-Heinemann, Newton, MA, USA,

v. 37, n. 6, p. 585–592, maio 2005. ISSN 0010-4485. Disponível em: <http://dx.doi.org/10.1016/j.cad.2004.11.002>.

CHOI, K.-J.; KO, H.-S. Stable but responsive cloth. In: *ACM SIGGRAPH 2005 Courses*. New York, NY, USA: ACM, 2005b. (SIGGRAPH '05). Disponível em: <http://doi.acm.org/10.1145/1198555.1198571>.

CLINE, M. B. *Rigid Body Simulation with Contact and Constraints*. Dissertação (Mestrado) — The University of Texas at Austin, 2002.

COHEN, J. D. et al. I-collide: An interactive and exact collision detection system for large-scale environments. In: *Symposium on Interactive 3D Graphics*. [s.n.], 1995. p. 189–196, 218. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.55.5123>.

COLLIER, J. R. et al. Drape prediction by means of finite-element analysis. *Journal of the Textile Institute*, v. 82, n. 1, p. 96–107, 1991. Disponível em: <http://www.tandfonline.com/doi/abs/10.1080/00405009108658741>.

CORDIER, F. *Real-time Animation of Dressed Virtual Humans*. Tese (Doutorado) — University of Geneva, 2004. Disponível em: <http://www.mage.fst.uha.fr/cordier/Home_files/Papers/Thesis.pdf>.

CORDIER, F.; MAGNENAT-THALMANN, N. Real-time animation of dressed virtual humans. *Computer Graphics Forum*, Blackwell Publishing, Inc, v. 21, n. 3, p. 327–335, 2002. ISSN 1467-8659. Disponível em: <http://dx.doi.org/10.1111/1467-8659.t01-1-00592>.

DECORO, C.; PAJAROLA, R. Xfastmesh: fast view-dependent meshing from external memory. In: *Proceedings of the conference on Visualization '02*. Washington, DC, USA: IEEE Computer Society, 2002. (VIS '02), p. 363–370. ISBN 0-7803-7498-3.

EISCHEN, J. W.; DENG, S.; CLAPP, T. G. Finite-element modeling and control of flexible fabric parts. *IEEE Comput. Graph. Appl.*, IEEE Computer Society Press, Los Alamitos, CA, USA, v. 16, n. 5, p. 71–80, set. 1996. ISSN 0272-1716. Disponível em: <http://dx.doi.org/10.1109/38.536277>.

ERICSON, C. *Real-Time Collision Detection*. Morgan Kaufmann, 2005. Hardcover. ISBN 1558607323. Disponível em: <http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/1558607323>.

ERLEBEN, K. *Stable, Robust, and Versatile Multibody Dynamics Animation*. Tese (Doutorado) — University of Copenhagen, Denmark, 2004. Disponível em: <http://image.diku.dk/kenny/download/erleben.05.thesis.pdf>.

FRISKEN, S. F. et al. Adaptively sampled distance fields: A general representation of shape for computer graphics. In: *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 2000. (SIGGRAPH '00), p. 249–254. ISBN 1-58113-208-5. Disponível em: <http://dx.doi.org/10.1145/344779.344899>.

FUHRMANN, A.; SOBOTKA, G.; GROSS, C. Distance fields for rapid collision detection in physically based modeling. *Proceedings of GraphiCon 2003*, 2003. Disponível em: <http://www.graphicon.ru/html/2003/Proceedings/Technical/paper495.pdf>.

GILBERT, E. G.; JOHNSON, D. W.; KEERTHI, S. S. A fast procedure for computing the distance between complex objects in three-dimensional space. *Robotics and Automation, IEEE Journal of*, v. 4, n. 2, p. 193–203, 1988. Disponível em: <http://dx.doi.org/10.1109/56.2083>.

KIRKPATRICK, D.; KLAWE, M.; TARJAN, R. Polygon triangulation inO (n log logn) time with simple data structures. *Discrete & Computational ...*, 1992. Disponível em: <http://link.springer.com/article/10.1007/BF02187846>.

KITCHENHAM, B. Procedures for performing systematic reviews. *Technical Report TR/SE-0401*, v. 1, p. 1–21, 2004.

MACRI, D. P. Real-Time Cloth. *Game Developers Conference Proceedings*, 2000.

METAAPHANON, N. et al. Simulation of tearing cloth with frayed edges. *Computer Graphics Forum*, Blackwell Publishing Ltd,

v. 28, n. 7, p. 1837–1844, 2009. ISSN 1467-8659. Disponível em: <http://dx.doi.org/10.1111/j.1467-8659.2009.01561.x>.

MIRTICH, B. V. *Impulse-based Dynamic Simulation of Rigid Body Systems*. Tese (Doutorado) — University of California at Berkeley, 1996. Disponível em: <http://www.kuffner.org/james/software/dynamics/mirtich/mirtichThesis.pdf>.

MüLLER, M. Hierarchical position based dynamics. In: *VRIPHYS'08*. [s.n.], 2008. p. 1–10. Disponível em: <http://matthiasmueller.info/publications/hpbd.pdf>.

MüLLER, M.; CHENTANEZ, N. Wrinkle meshes. In: *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2010. (SCA '10), p. 85–92. Disponível em: <http://dl.acm.org/citation.cfm?id=1921427.1921441>.

MüLLER, M. et al. Position based dynamics. *J. Vis. Comun. Image Represent.*, Academic Press, Inc., Orlando, FL, USA, v. 18, n. 2, p. 109–118, abr. 2007. ISSN 1047-3203. Disponível em: <http://dx.doi.org/10.1016/j.jvcir.2007.01.005>.

MüLLER, M. et al. Real time physics: class notes. *ACM SIGGRAPH 2008 classes*, 2008. Disponível em: <http://dl.acm.org/citation.cfm?id=1401245>.

NEALEN, A. et al. Physically based deformable models in computer graphics. *Computer Graphics Forum*, Blackwell Publishing, v. 25, n. 4, p. 809–836, dez. 2006. ISSN 0167-7055. Disponível em: <http://dx.doi.org/10.1111/j.1467-8659.2006.01000.x>.

NGUYEN, H. *GPU Gems 3*. Addison-Wesley Professional, 2007. Hardcover. ISBN 0321515269. Disponível em: <http://developer.nvidia.com/object/gpu-gems-3.html>.

NOBREGA, T. de H. C.; CARVALHO, D. D. B.; WANGENHEIM, A. von. Using metaprogramed functors to implement double-dispatch collision handling. In: *XXI Brazilian Symposium on Computer Graphics and Image Processing*. [s.n.], 2008. Disponível em: <http://www.gpec.ucdb.br/sibgrapi2008/posters/posters/47829.pdf>.

PLATT, J. C.; BARR, A. H. Constraints methods for flexible models. *SIGGRAPH Comput. Graph.*, ACM, New York, NY, USA, v. 22, n. 4, p. 279–288, jun. 1988. ISSN 0097-8930. Disponível em: <http://doi.acm.org/10.1145/378456.378524>.

PRITCHARD, D. Implementing Baraff & Witkin's Cloth Simulation'. n. May 2003, 2003. Disponível em: <http://davidpritchard.org/freecloth/docs/report.pdf http://www.davidpritchard.org/freecloth/docs/report.pdf>.

PROVOT, X. Deformation constraints in a Mass-Spring model to describe rigid cloth behavior. In: DAVIS, W. A.; PRUSINKIEWICZ, P. (Ed.). *Graphics Interface '95*. Canadian Human-Computer Communications Society, 1995. p. 147–154. Disponível em: <http://citeseer.ist.psu.edu/provot96deformation.html>.

PROVOT, X. Collision and self-collision handling in cloth model dedicated to design garments. In: THALMANN, D.; PANNE, M. (Ed.). *Computer Animation and Simulation '97*. [S.l.]: Springer Vienna, 1997, (Eurographics). p. 177–189. ISBN 978-3-211-83048-2.

RUPPERT, J. A Delaunay refinement algorithm for quality 2-dimensional mesh generation. *Journal of algorithms*, p. 1–47, 1995. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0196677485710218>.

SCHMITT, N. et al. Multilevel cloth simulation using gpu surface sampling. In: *Virtual Reality Interactions and Physical Simulations (VRIPhys)*. Lille, France: Eurographics Association, 2013.

SHEWCHUK, J. R. *An Introduction to the Conjugate Gradient Method Without the Agonizing Pain*. Pittsburgh, PA, USA, 1994.

SHEWCHUK, J. R. Triangle: Engineering a 2d quality mesh generator and delaunay triangulator. In: *Selected papers from the Workshop on Applied Computational Geormetry, Towards Geometric Engineering*. London, UK, UK: Springer-Verlag, 1996. (FCRC '96/WACG '96), p. 203–222. ISBN 3-540-61785-X. Disponível em: <http://dl.acm.org/citation.cfm?id=645908.673287>.

SOUZA, M. S. Animação baseada em física: Desenvolvimento de um simulador de corpos rígidos para aplicações interativas. 2011. Disponível em: <http://www.inf.ufsc.br/~sms/tcc/docs/tcc_planck_v16.pdf>.

TANG, M. et al. A gpu-based streaming algorithm for high-resolution cloth simulation. *Computer Graphics Forum*, v. 32, n. 7, p. 21–30, 2013.

TERZOPOULOS, D. et al. Elastically deformable models. *SIGGRAPH Comput. Graph.*, ACM, New York, NY, USA, v. 21, n. 4, p. 205–214, ago. 1987. ISSN 0097-8930. Disponível em: <http://doi.acm.org/10.1145/37402.37427>.

TESCHNER, M. et al. Collision detection for deformable objects. *Computer Graphics Forum*, Blackwell Publishing Ltd., v. 24, n. 1, p. 61–81, 2005. ISSN 1467-8659. Disponível em: <http://dx.doi.org/10.1111/j.1467-8659.2005.00829.x>.

VOLINO, P.; THALMANN, N. M. Collision and self-collision detection: Efficient and robust solutions for highly deformable surfaces. In: *In*. [S.l.]: Springer Verlag, 1995. p. 55–65.